Novell
Policies in iManager

3.6

www.novell.com

POLICIES IN IMANAGER

April 30, 2008

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 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 information on creating and managing policies in iManager.

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 iManager*, visit the Identity Manager Documentation Web site (http://www.novell.com/documentation/idm35).

Additional Documentation

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

For documentation on Novell iManager, see the Novell iManager Documentation Web site (http://www.novell.com/documentation/imanager26/index.html).

For documentation on Designer see, the *Designer 3.0 for Identity Manager 3.6 Administration Guide*.

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 synchronized between the Identity Vault and the remote data store. The policies are stored in policy sets. Identity Manager installs iManager plug-ins that allow you to create and manage policies.

In order to access the objects that are used in policies, see "iManager Navigation" on page 269.

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

- Policies are made up of rules.
- A rule is a set of conditions (see Chapter 9, "Conditions," on page 91) that must be met before a defined action (see Chapter 10, "Actions," on page 135) occurs.
- Actions can have dynamic arguments that derive from tokens that are expanded at run time.
- Tokens are divided into two classifications: nouns and verbs.
  - Noun tokens (see Chapter 11, "Noun Tokens," on page 209) expand to values that are derived from the current operation, the source or destination data stores, or some external source.
  - Verb tokens (see Chapter 12, "Verb Tokens," on page 247) modify the concatenated results of other tokens that are subordinate to them.
- Regular expressions (see "Regular Expressions" in *Understanding Policies for Identity Manager 3.6*) and XPath 1.0 expressions (see "XPath 1.0 Expressions" in *Understanding Policies for Identity Manager 3.6*) 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 Novell's `nds.dtd`; for more information, see "NDS DTD" in the *Identity Manager 3.6 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.

For more information on policies and policy types, see *Understanding Policies for Identity Manager 3.5.1* (http://www.novell.com/documentation/idm35/policy/data/policytypesoverview.html).

The following sections explain how to create and use policies.

- Chapter 2, "Managing Policies with Policy Builder," on page 15
- Chapter 3, "Using Additional Builders," on page 25
- Chapter 4, "Defining Schema Mapping Policies," on page 35
- Chapter 5, "Controlling the Flow of Objects with the Filter," on page 41

- Chapter 6, “Using Predefined Rules,” on page 47
- Chapter 7, “Storing Information in Resource Objects,” on page 73
- Chapter 8, “Using ECMAScript in Policies,” on page 83

This guide also contains a detailed reference section for all of the elements in DirXML® Script. For more information on DirXML Script, see “DirXML Script DTD” in *Identity Manager 3.6 DTD Reference*.

- Chapter 9, “Conditions,” on page 91
- Chapter 10, “Actions,” on page 135
- Chapter 11, “Noun Tokens,” on page 209
- Chapter 12, “Verb Tokens,” on page 247

Managing Policies with Policy Builder

2

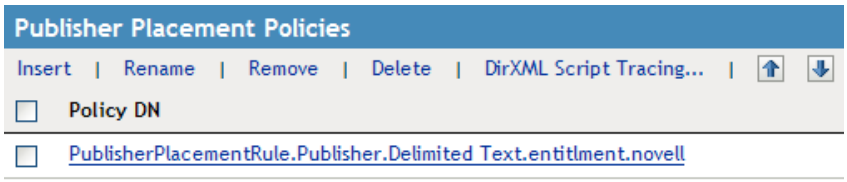
The Policy Builder is a complete graphical interface for creating and managing the policies that define the exchange of data between connected systems.

2.1 Accessing the Policy Builder

1 Access the Identity Manager Driver Overview by following the steps in “Accessing the Identity Manager Driver Overview Page” on page 270.

Ensure that the driver that is displayed in the Identity Manager Overview is the driver for which you want to manage policies.

2 Click the desired policy set, then click the policy you want to edit to open the Policy Builder.



2.2 Creating a Policy

A policy can be created in a driver or in a library object.

2.2.1 Creating a Policy in a Driver

- "Using an Existing Policy to Create a Policy" on page 17

Creating a New Policy

1 Open the Identity Manager Driver Overview for the driver you want to manage.

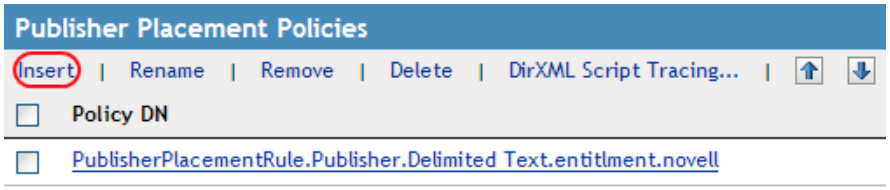
For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click a policy set icon.

represents an undefined policy.

represents a defined policy.

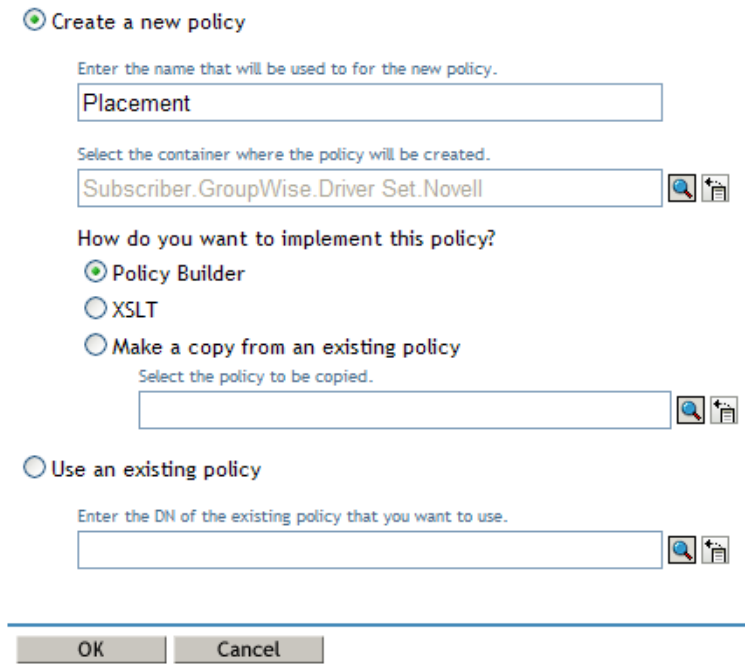
3 Click *Insert*.



4 Select *Create a new policy*.

5 Specify a name for the new policy.

6 Select how to implement the policy, then click *OK*.



- If you select *Policy Builder*, the Policy Builder is launched. To define one or more rules for this policy, click *Append New Rule*, then follow the instructions in Section 2.3, "Defining Individual Rules within a Policy," on page 18.

- If you select *XSLT*, the XML editor is launched. To define the policy with XSLT, see "Defining Policies by Using XSLT Style Sheets" in *Understanding Policies for Identity Manager 3.6*.
- If you select *Make a copy from an existing policy*, browse to and select the policy to copy.

Using an Existing Policy to Create a Policy

1. Open the Identity Manager Driver Overview for the driver you want to manage.

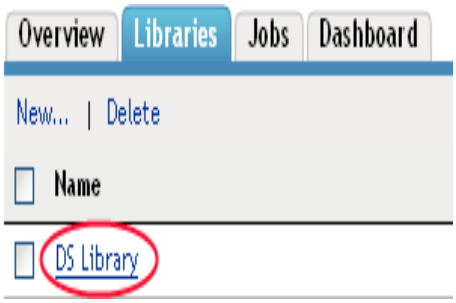
   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set icon.

   represents an undefined policy.

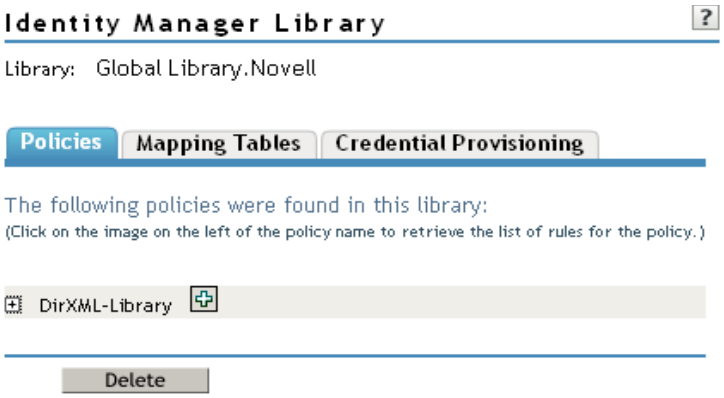
   represents a defined policy.
3. Click *Insert*.
4. Select *Use an existing policy*, then browse to and select the existing policy you want to use.
5. Click *OK*.

2.2.2 Creating a Policy in a Library

1. Access the Identity Manager Driver Set Overview by following the steps in "Accessing the Identity Manager Driver Set Overview Page" on page 269.
2. Click the *Libraries* tab.
3. Click the library you want to add a policy to.

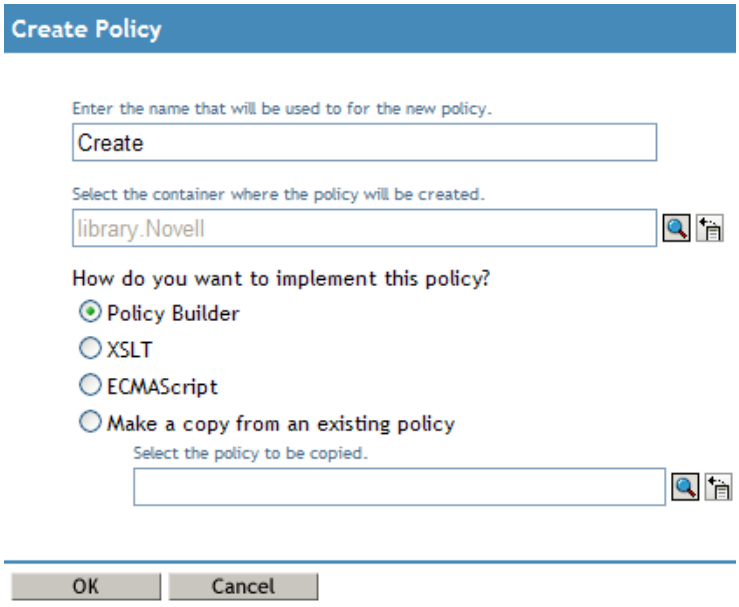


4. Click the plus icon to add a policy to the library.



5 Specify a name for the policy.

6 Select how to implement the policy, then click *OK*.



- If you select *Policy Builder*, *XSLT*, or *ECMAScript*, the object is created and displayed in the library. Each object must be edited to add the policy information into the object.
- If you select *Make a copy from an existing policy*, browse to and select the policy to store in the library.

2.3 Defining Individual Rules within a Policy

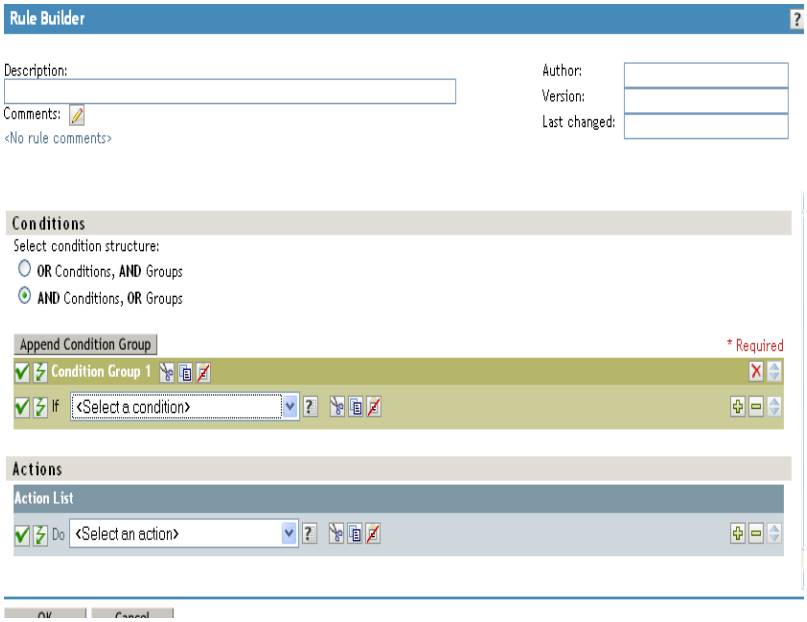
Rules are defined in the Rule Builder window of the Policy Builder. To access the Rule Builder window:

1 Click the library that contains the policy of the rules you want to define.

2 Click on the policy.

3 Click *Append New Rule*.

Figure 2-1 *Rule Builder Window of the Policy Builder*



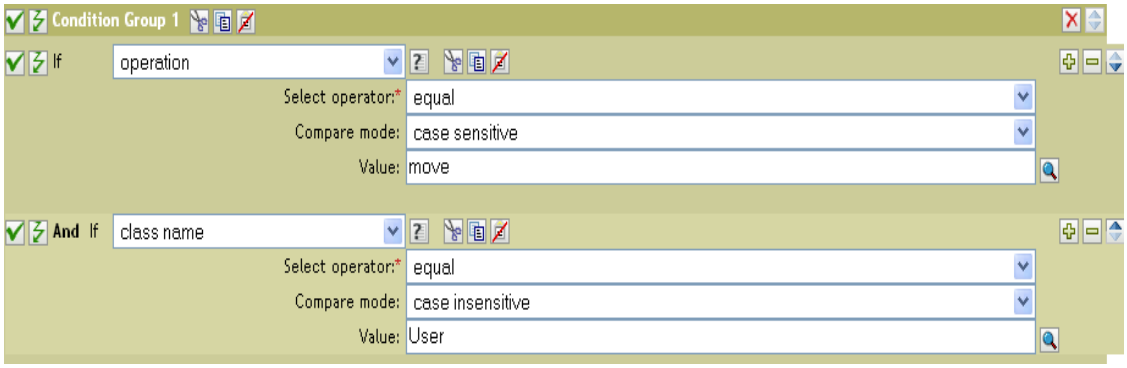
The Rule Builder interface enables you to quickly create and modify rules using intelligent drop-down menus.

In the Rule Builder, you define a set of conditions that must be met before a defined action occurs.

For example, if you need to create a rule that disallows any new objects from being added to your environment, you might define this rule to indicate that when an add operation occurs, veto the operation.

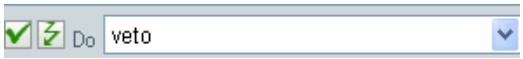
To implement this logic in the Rule Builder, you could select the following condition:

Figure 2-2 *Move User Condition in the Rule Builder Interface*



And the following action:

Figure 2-3 *Veto Action in the Rule Builder Interface*

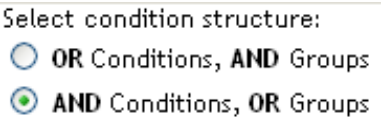


See Chapter 9, “Conditions,” on page 91 and Chapter 10, “Actions,” on page 135 for a detailed reference on the conditions and actions available in the Rule Builder.

Tips

To create more complex conditions, you can join conditions and groups of conditions with and/or statements. You can modify the way these are joined by selecting the condition structure:

Figure 2-4 *Condition Structure Radio Buttons*



- **Browse:** Click the icon to see a list of values for a field. In the example above, this icon opens a list of valid class names.
- **Argument Builder:** Click the icon to use the Argument Builder interface to construct an argument.
- **Enable/Disable Policy, Rule, Condition or Action:** Click the icon to disable a policy, rule, condition, or action. Click the icon to re-enable it.
- **Enable/Disable Policy Tracing:** Click the icon to disable tracing on the policy. Click the icon to re-enable tracing of the policy.
- **Comment:** Click the icon to add a comment to a policy or rule. Comments are stored directly on the policy or rule, and can be as long as necessary.
- **Cut/Copy/Paste:** Use the Cut/Copy/Paste icons to use the Policy Builder clipboard. The Paste icon is disabled if the current content on the clipboard is invalid at that location.
- **Conditions:** Use the icons to add, remove, and position conditions.
- **Add Condition Groups:** Use the Append Condition Group button to add condition groups.
- **Remove and Position Condition Groups:** Use the icons to remove and position condition groups.

2.4 Creating Arguments within a Rule

The Argument Builder provides a dynamic graphical interface that enables you to construct complex argument expressions for use within the Rule Builder. To access the Argument Builder, see “Argument Builder” on page 26.

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

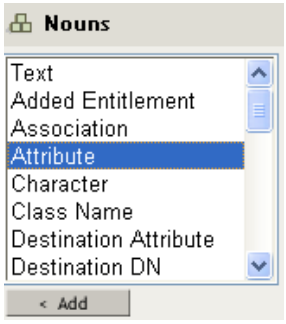
Tokens are divided into two classifications: nouns and verbs. Noun tokens expand to values that are derived from the current operation, the source or destination data stores, or some external source. Verb tokens modify the results of other tokens that are subordinate to them.

Figure 2-5 *Default Argument Builder Interface*

To define an expression, select one or more noun 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.

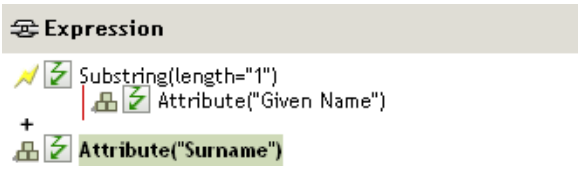
For example, if you want the argument set to an attribute value:

1 In the Argument Builder, select *Attribute* from the list of noun tokens, then click *Add*.



2 Browse to and select the attribute name in the editor.

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



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

See Chapter 11, "Noun Tokens," on page 209 and Chapter 12, "Verb Tokens," on page 247 for a detailed reference on the noun and verb tokens. See Section 3.2, "Argument Builder," on page 26 for more information on the Argument Builder.

2.5 Modifying a Policy

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set icon.
3. Click the name of the policy you want to modify.

   The Policy Builder is launched.
4. Make the desired modifications, then click *OK*.

2.6 Removing a Policy

The Remove option removes the policy from the selected Policy Set but doesn't delete the policy.

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set icon, select the policy you want to remove, then click *Remove*.

To view a policy that is not associated with a policy set:

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click *Advanced* > *Show All Policies*.

To add the removed policy back to the policy set:

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set icon.
3. Click *Insert*.
4. Select *Use an existing policy*, then click the browse button.
5. Browse to the policy you want to add.

   Make sure you are in the proper container to see the policy.
6. Click *OK*.
7. Click *Close*.

2.7 Renaming a Policy

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set icon.
3. Select the policy you want to rename.
4. Click *Rename* and rename the policy.
5. Click *OK*.
6. Click *Close*.

2.8 Deleting a Policy

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set icon.
3. Select the policy you want to delete, then click *Delete*.

2.9 Exporting a Policy to an XML File

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set icon.
3. Click the name of a policy.
4. Click the *Save As* button, then select a location to save the DirXML® Script XML file.
5. Click *Save.*

2.10 Importing a Policy from an XML File

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set icon.
3. Click the name of a policy.
4. Click the *Insert* button, then select *Import an XML file containing DirXML Script.*
5. Browse to and select the policy file to import, then click *OK*.

2.11 Creating a Policy Reference

A policy reference enables you to create a single policy, and reference it in multiple locations. If you have a policy that is used by more than one driver or policy, creating a reference simplifies management of this policy.

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set icon.
3. Click the name of a policy.
4. Click the *Insert* button, and select *Append a reference to a policy containing DirXML Script*.
5. Browse to and select the policy object to reference, then click *OK*.

Using Additional Builders

3

Although you define most arguments by using the Argument Builder (see Section 2.4, “Creating Arguments within a Rule,” on page 20), 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:

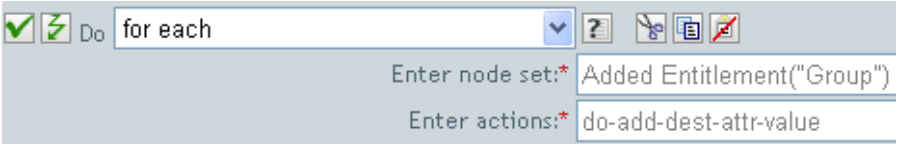
- Section 3.1, “Argument Actions Builder,” on page 25
- Section 3.2, “Argument Builder,” on page 26
- Section 3.3, “Match Attribute Builder,” on page 30
- Section 3.4, “Action Argument Component Builder,” on page 31
- Section 3.5, “Argument Value List Builder,” on page 31
- Section 3.6, “String Builder,” on page 32
- Section 3.7, “Condition Argument Component Builder,” on page 33

3.1 Argument Actions Builder

The Argument Actions Builder enables you to set the action that is required by the For Each action and the Implement Entitlement action.

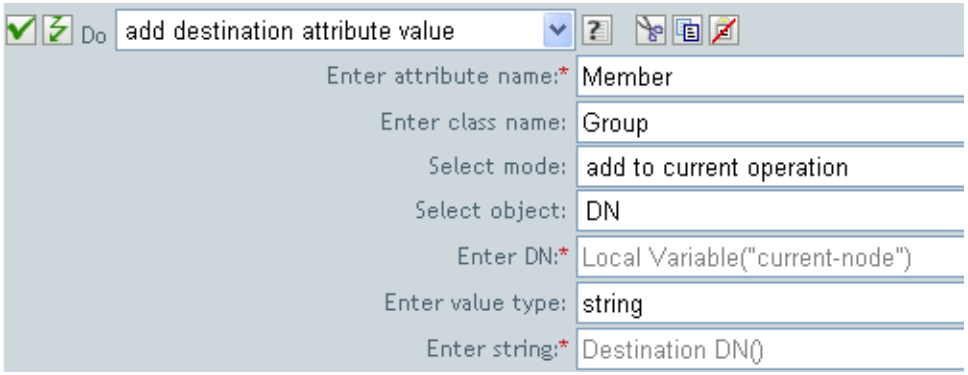
In the following example, the add destination attribute value action is performed for each Group entitlement that is being added in the current operation.

Figure 3-1 *Action For Each*



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

Figure 3-2 *Action Add Destination Attribute Value*



3.2 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. For more information on noun tokens, see Chapter 11, “Noun Tokens,” on page 209.
- **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. For more information on verb tokens, see Chapter 12, “Verb Tokens,” on page 247.
- **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:** Use the Editor pane to provide the values for the nouns and the verbs.

Figure 3-3 *Argument Builder*

Launch the Argument Builder from the following actions by clicking the *Edit Arguments* icon.

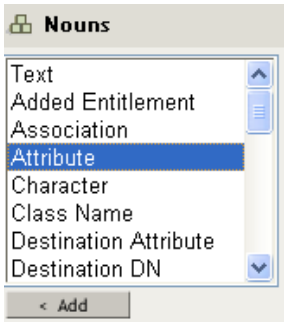
- Add Association
- Add Destination Attribute Value
- Add Destination Object
- Add Source Attribute Value
- Append XML Text
- Clear Destination Attribute Value when the selected object is DN or Association.
- Clear Source Attribute Value when the selected object is DN or Association.
- Delete Destination Object when the selected object is DN or Association.
- Delete Source Object when the selected object is DN or Association.

- Find Matching Object
- For Each
- Move Destination Object
- Move Source Object
- Reformat Operation Attribute
- Remove Association
- Remove Destination Attribute Value
- Remove Source Attribute Value
- Rename Destination Object when the selected object is DN or Association and Enter String.
- Rename Source Object when the selected object is DN or Association and Enter String.
- Set Destination Attribute Value when the selected object is DN or Association, and the Enter Value type is not structured.
- Set Destination Password
- Set Local Variable
- Set Operation Association
- Set Operation Class Name
- Set Operation Destination DN
- Set Operation Property
- Set Operation Source DN
- Set Operation Template DN
- Set Source Attribute Value
- Set Source Password
- Set XML Attribute
- Status
- Trace Message

To define an expression, select one or more nouns (values, objects, variables, etc.), and combine them with verbs (substring, escape, uppercase and lowercase) to construct arguments.

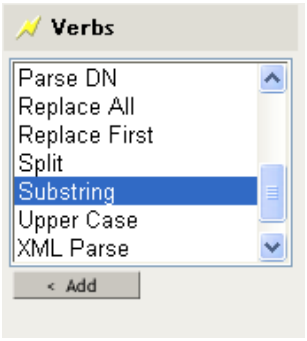
The following example creates an argument for a username from the first letter of the first name and the entire last name:

1 Select *Attribute* from the list of nouns, then click *Add*.



2 Specify or select the Given Name attribute.

3 Select *Substring* from the list of verbs, then click *Add*.



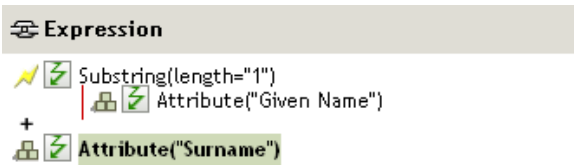
4 Type 1 in the *Length* field.

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

6 Select *Attribute* from the list of nouns, then click *Add*.

7 Specify or browse to the *Surname* attribute.

8 Select the Surname attribute, then click the *Move Down* icon twice.



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

9 Click *OK* to save the argument.

3.2.1 Argument Builder Tips

- Use the Cut/Copy/Paste icons  to use the Policy Builder clipboard. The *Paste* icon is disabled if the current content on the clipboard is invalid at that location.
- Use the Move Up/Move Down/Remove icons  to reposition or remove tokens in the argument.
- Use the update the expression panel link to refresh the Argument Builder interface. The interface is refreshed automatically whenever you add or modify a token.

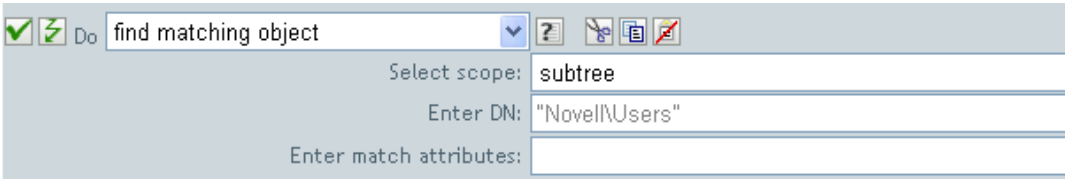
3.3 Match Attribute Builder

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

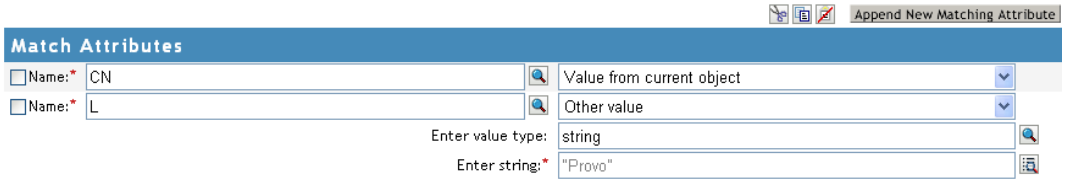
The following example matches users if the users are based in Provo and have a unique CN attribute:

1. In the Rule Builder, select *find matching object*.

   For information on accessing the Rule Builder, see "Defining Individual Rules within a Policy" on page 18.
2. Select the Scope of the search as *subtree*.
3. Browse to and select the location to search. In this example, it is the Users container.
4. Click the icon next to the *Enter Match Attributes* field to launch the Match Attribute Builder.



5. Click *Append New Matching Attribute* to add an attribute to match.
6. Specify the CN attribute in the *Name* field.
7. Select *Value from current object* to see if there are any other users with the same CN attribute.
8. Click *Append New Matching Attribute* to add another attribute to match.
9. Specify the L attribute in the *Name* field.
10. Select *Other Value*, then specify Provo as the value.



11. Click *OK*.

3.3.1 Match Attribute Builder Tips

Use the Cut/Copy/Paste icons  to use the Policy Builder clipboard. The *Paste* icon is disabled if the current content on the clipboard is invalid at that location.

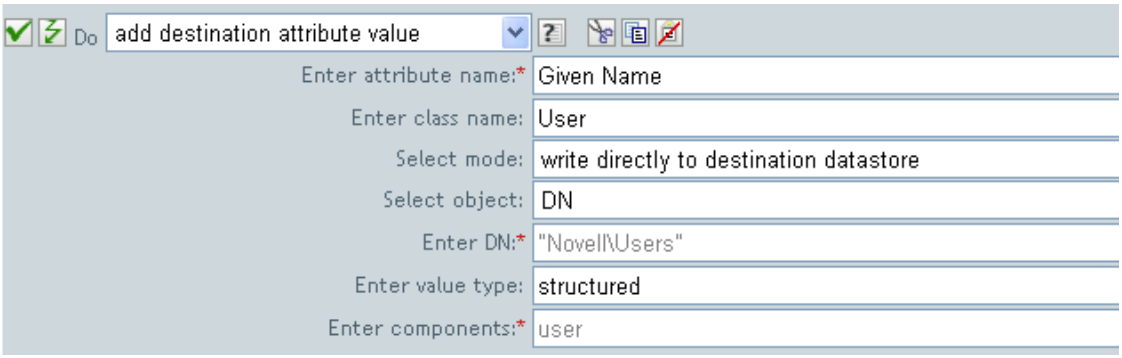
3.4 Action Argument Component Builder

In the Rule Builder, launch the Action Argument Component Builder by selecting the following actions when the *Enter Value Type* selection is set to Structured.

For information on accessing the Rule Builder, see "Defining Individual Rules within a Policy" on page 18.

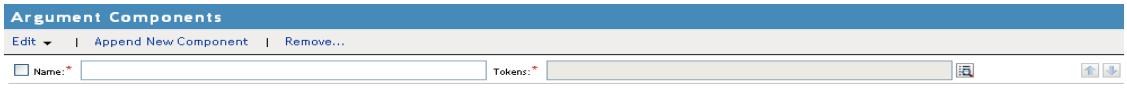
- Add Destination Attribute Value (page 138)
- Add Source Attribute Value (page 142)
- Reformat Operation Attribute (page 169)
- Remove Destination Attribute Value (page 171)
- Remove Source Attribute Value (page 172)
- Set Default Attribute Value (page 180)
- Set Source Attribute Value (page 192)

Figure 3-4 *Action Value Type Field Set to Structured*



After the value type is set to structured, click the *Edit components* icon.

Figure 3-5 *Action Argument Component Builder*



The Action Argument Component Builder is launched and the action can be constructed.

3.5 Argument Value List Builder

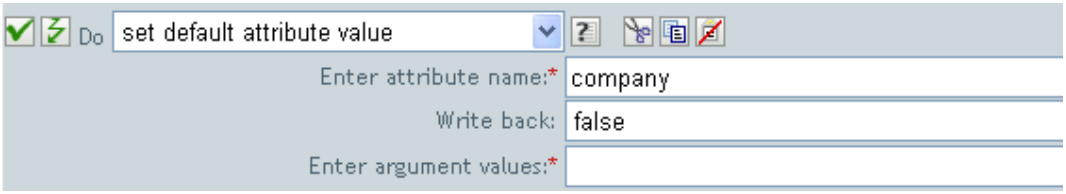
The Argument Value List Builder enables you to construct default argument values for the Set Default Attribute Value action.

For example, if you want to set a default company name:

1. In the Rule Builder, select *set default attribute value* from the list of actions.

   For information on accessing the Rule Builder, see "Defining Individual Rules within a Policy" on page 18.

2. Browse to and select the company attribute.



3. Click the *Edit the value list* icon to create the company name.
4. Click *Append New Value* in the Argument Value List Builder.
5. Specify the name of the company.

   For this example, the company name is Digital Airlines.

6. Click *OK* twice.

3.5.1 Argument Value List Builder Tips

Use the Cut/Copy/Paste icons to use the Policy Builder clipboard. The *Paste* icon is disabled if the current content on the clipboard is invalid at that location.

3.6 String Builder

The String Builder enables you to construct name/value pairs for use in certain actions such as Generate Event, Send Email, and Send Email from Template.

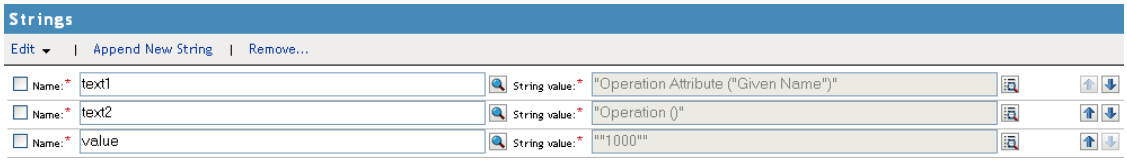
You can access the String Builder by clicking the *Edit the strings* icon located in the Action List section of the Rule Builder. For information on accessing the Rule Builder, see "Defining Individual Rules within a Policy" on page 18.

For the *Generate Event* action, the string names correspond to the custom value fields you can provide with an event:

- target
- target-type
- subTarget
- text1
- text2
- text3

- value
- value3
- data
- data-type

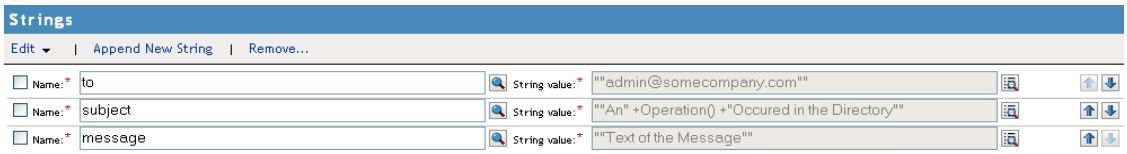
Figure 3-6 *String Builder*



For the *Send Email* action, the string names correspond to the elements of the e-mail:

- to
- cc
- bcc
- from
- reply-to
- subject
- message
- encoding
- custom-smpt-header

Figure 3-7 *Send Mail Action*



For the *Send Email* from Template action, the named strings correspond to the elements of the e-mail in the template:

- to
- cc
- bcc
- reply-to
- encoding
- custom-smtp-header

3.7 Condition Argument Component Builder

Launch the Condition Argument Component Builder by clicking the *Edit arguments* icon in the Rule Builder. For information on accessing the Rule Builder, see "Defining Individual Rules within a Policy" on page 18.

In order to see the icon, you must select the Structured selection for Mode with the following conditions:

- If Attribute
- If Destination Attribute
- If Source Attribute

Figure 3-8 *Structured Option*

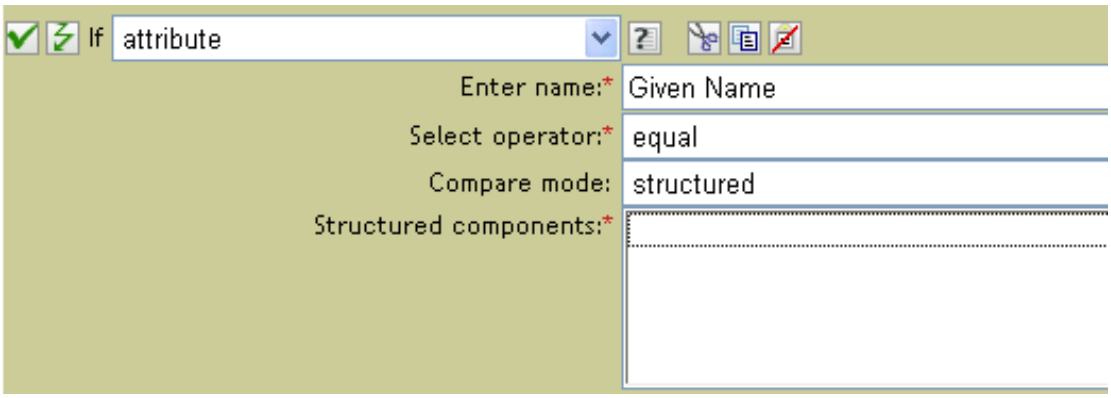
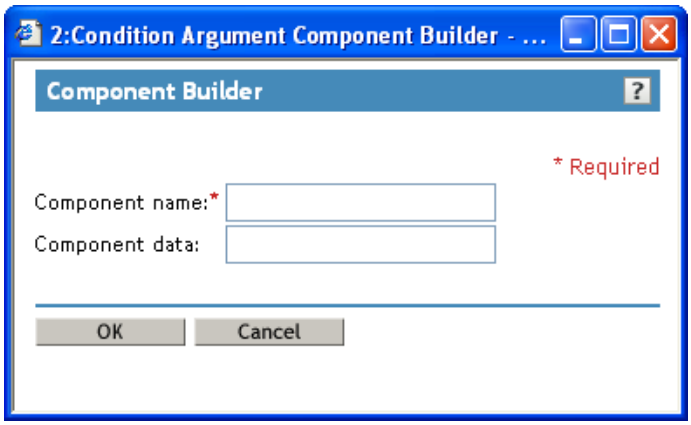


Figure 3-9 *Condition Argument Component Builder*



Defining Schema Mapping Policies

4

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.

4.1 Accessing Schema Mapping Policies

1 To access a Schema Mapping Policy, navigate to the Identity Manager Driver Overview page.

For instructions on how to access the Identity Manager Driver Overview page, see “Accessing the Identity Manager Driver Overview Page” on page 270.

2 In the Identity Manager Driver Overview page, click the Schema Mapping Policy set.

The Schema Mapping Policies are displayed.

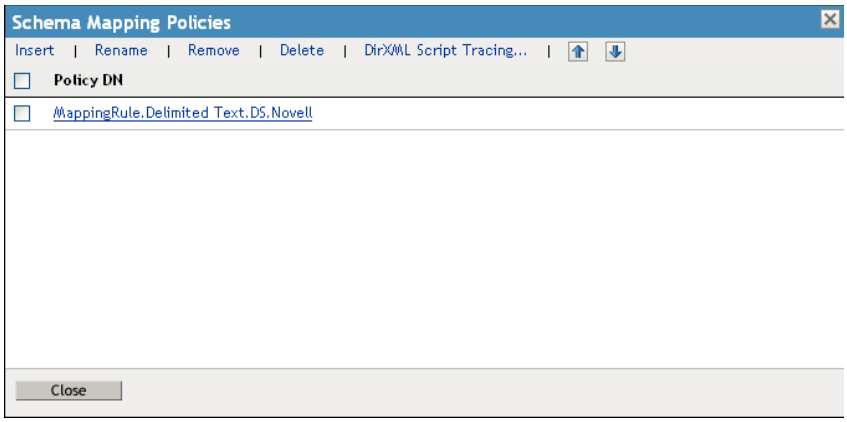
4.2 Editing the Schema Mapping Policy

There are two different parts to editing a Schema Mapping policy. First, you edit the placement of the policies in the policy set. Second, you edit the policy itself through the Schema Map editor.

4.2.1 Placement of the Policies

1 In the Identity Manager Driver Overview page, click the Schema Mapping Policy to bring up the Schema Mapping Policies window.

For instructions on how to access the Identity Manager Driver Overview page, see “Accessing the Identity Manager Driver Overview Page” on page 270.



The options in this window allow you to position the policy you are currently working with. The following table explains each of the options:

Option	Description
Insert	Inserts a new or an existing policy into the policies listed.
Rename	Renames the selected policy.
Remove	Removes the selected policy without deleting the policy from the policy set.
Delete	Deletes the selected policy.
DirXML Script Tracing	Turns DirXML® Script tracing or DirXML Rule tracing on or off.
Move Policy Up	Moves the selected policy up if there is more than one policy.
Move Policy Down	Moves the selected policy down if there is more than one policy.
Policy DN	Simultaneously selects all policies.

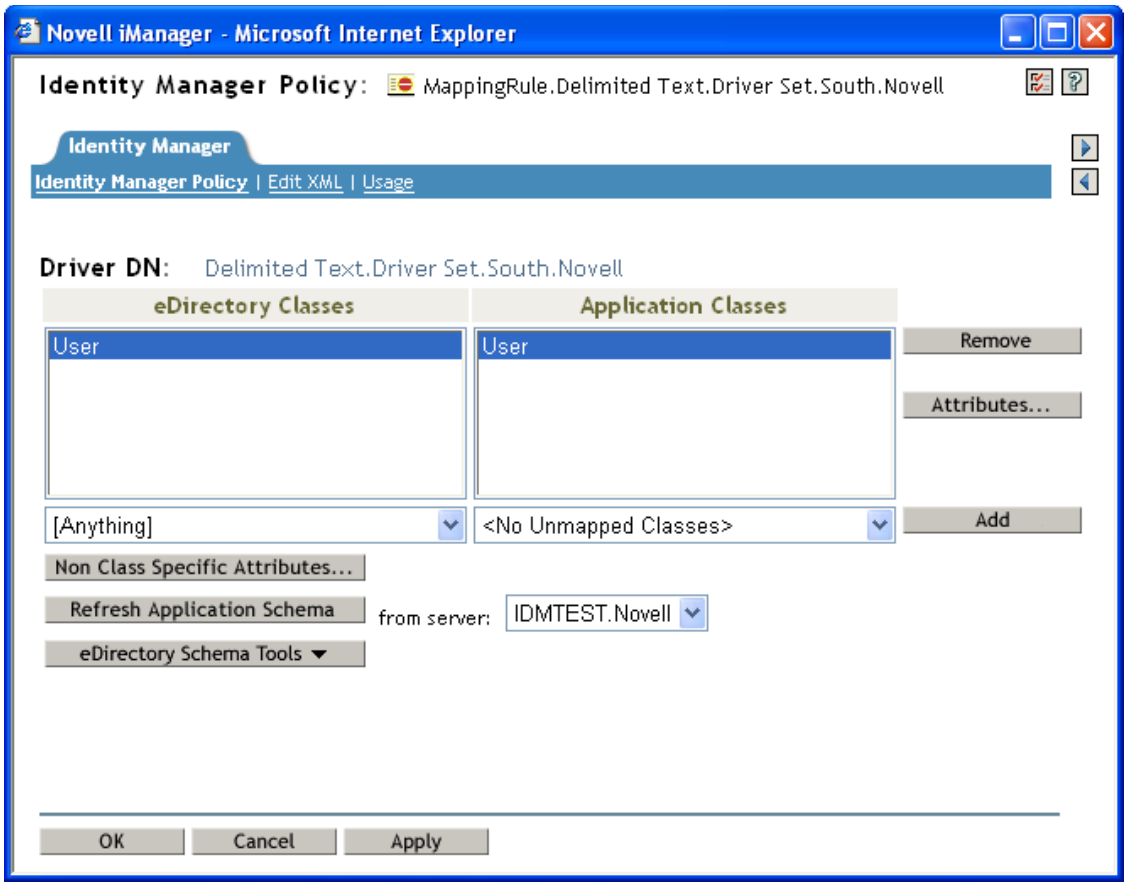
4.2.2 Schema Map Editor

The Schema Map editor is a complete graphical interface for creating and managing the schema mapping policies. The Schema Map editor creates a policy by using XML.

To access the Schema Map Editor:

1. On the Identity Manager Driver Overview page, click the Schema Mapping Policy set.

   For instructions on how to access the Identity Manager Driver Overview page, see “Accessing the Identity Manager Driver Overview Page” on page 270.
2. Click the name of a policy.



The Schema Map editor has three tabs:

- “Identity Manager Policy” on page 37
- “Edit XML” on page 38
- “Usage” on page 39

Identity Manager Policy

Contains the most information and is where you edit the policy through the GUI interface.

Table 4-1 *Schema Map Editor Tasks*

Removing Classes and Attributes	Select the class or attribute you would like to remove, then click *Remove*.
Adding Classes	Select the eDirectory™ class from the drop-down list, then select the Application class from the drop-down list. With the items selected, click *Add*, then click *Apply* to save the change.

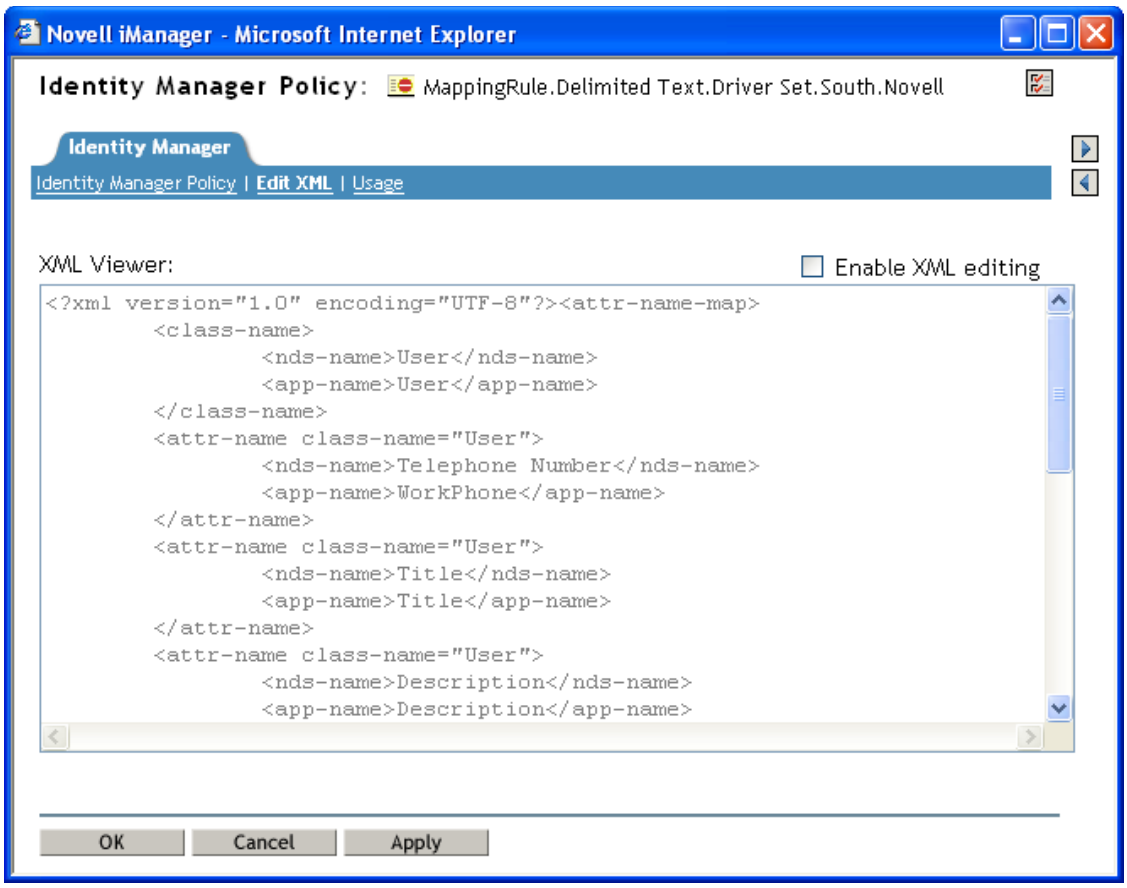
Adding Attributes	Select the class of the attribute you want to add, then click *Attribute*. Select the eDirectory attribute from the drop-down list, then select the Application attribute from the drop-down list. With the items selected, click *Add*, then click *OK* to save the changes.
Listing Non Specific Class Attributes	If there are attributes that are not associated with a class, click the *Non-specific Class Attributes* icon and all of these attributes are listed.
Refreshing Application Schema	If the schema has changed for the application, click the *Refresh Application Schema* icon. The wizard contacts the Connected System server to retrieve the new schema. After the schema has been updated, the schema is listed in the drop-down lists.
eDirectory Schema Tools	◆ **Add Attribute:** Adds an existing attribute to the selected class. ◆ **Create Attribute:** Creates a new attribute. ◆ **Create Class:** Creates a new class. ◆ **Delete Attribute:** Deletes the selected attribute. ◆ **Delete Class:** Deletes the selected class. ◆ **Refresh eDirectory Schema:** After making changes to the eDirectory schema, click *Refresh eDirectory Schema* to update the drop-down lists with the new information.

WARNING: Do not delete any classes or attributes that are being used in the Identity Vault. This can cause objects to become unknown.

Edit XML

Select *Enable XML editing* to edit the DirXML Script policy. Make the changes you desire to the DirXML Script, then click *Apply* to save the changes.

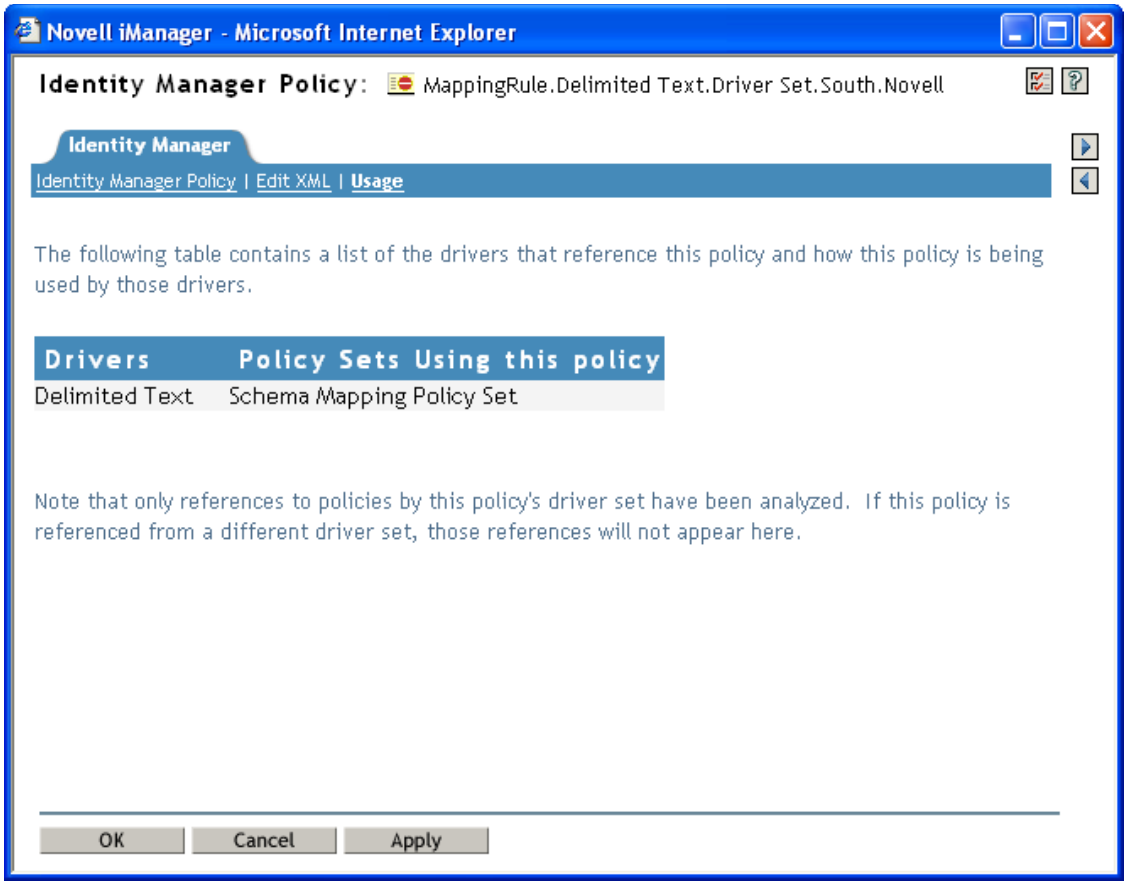
Figure 4-1 *Edit XML*



Usage

Shows you a list of the drivers that are currently referencing this policy. The list refers only to policies in this policy's driver set. If this policy is referenced from a different driver set, those references do not appear here.

Figure 4-2 *Usage*



Controlling the Flow of Objects with the Filter

5

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.

5.1 Accessing the Filter

1 To access the filter, navigate to the Identity Manager Driver Overview page.

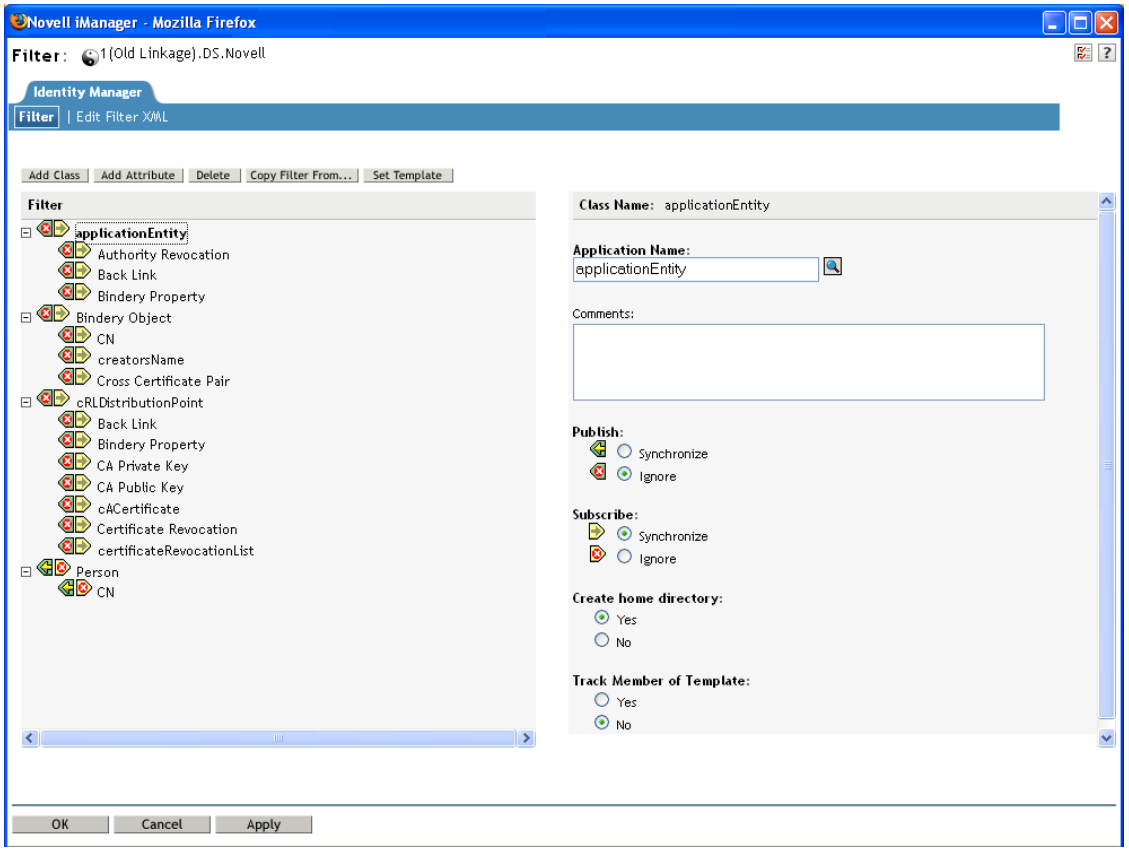
For instructions on how to access the Identity Manager Driver Overview page, see “Accessing the Identity Manager Driver Overview Page” on page 270.

2 Click the Filter icon on the Publisher or Subscriber channel. It is the same object.

5.2 Editing the Filter

The Filter editor gives you the options of editing how information is synchronized between the Identity Vault and the connected system.

Figure 5-1 *Filter Editor*



Here is a list of most common tasks when editing the filter:

- Section 5.2.1, “Removing a Class or an Attribute from the Filter,” on page 42
- Section 5.2.2, “Adding a Class,” on page 42
- Section 5.2.3, “Adding an Attribute,” on page 43
- Section 5.2.4, “Copying a Filter,” on page 43
- Section 5.2.5, “Setting a Template,” on page 43
- Section 5.2.6, “Changing the Filter Settings,” on page 43

5.2.1 Removing a Class or an Attribute from the Filter

1. Select the class or attribute, then click *Delete*.

5.2.2 Adding a Class

1. Click *Add Class*.
2. Click the type of class you want to add.
3. Change the options to synchronize the information.
4. Click *Apply*.

5.2.3 Adding an Attribute

1. Select the Class where you want the attribute to be added.
2. Click *Add Attribute*.
3. Select the attribute you want to add, then click *OK*.
4. Change the option to synchronize the information.
5. Click *Apply*.

5.2.4 Copying a Filter

You can copy the filter from an existing driver into the driver you are currently working on.

1. Click *Copy Filter From*.
2. Browse to and click the driver you want to copy the filter from.
3. Click *Apply* or *OK*.

5.2.5 Setting a Template

You can set the default values for an attribute you add to the filter.

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

You can change the values of the attributes after they have been created.

5.2.6 Changing the Filter Settings

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

1. In the Filter editor, select a class.
2. Change the filter settings for the selected class.

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	◆ **Yes:** Automatically creates home directories. ◆ **No:** Does not create home directories.

Options	Definitions
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.

3 Select an attribute.

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 not reported or 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.)
Subscribe	◆ **Synchronize:** Changes to this object are reported and automatically synchronized. ◆ **Ignore:** Changes to this object are not reported or 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.)

Options	Definitions
Merge Authority	◆ **Default:** If an attribute is not being synchronized in either channel, no merging occurs. If an attribute is being synchronized in one channel and not the other, then all existing values on the destination for that channel are removed and replaced with the values from the source for that channel. If the source has multiple values and the destination can only accommodate a single value, then only one of the values is used on the destination side. If an attribute is being synchronized in both channels and both sides can accommodate only a single value, the connected application acquires the Identity Vault values unless there is no value in the Identity Vault. If this is the case, the Identity Vault acquires the values from the connected application (if any). If an attribute is being synchronized in both channels and only one side can accommodate multiple values, the single-valued side's value is added to the multi-valued side if it is not already there. If there is no value on the single side, you can choose the value to add to the single side. This is always valid behavior. ◆ **Identity Vault:** Behaves the same way as the default behavior if the attribute is being synchronized on the Subscriber channel and not on the Publisher channel. This is valid behavior when synchronizing on the Subscriber channel. ◆ **Application:** Behaves the same as the default behavior if the attribute is being synchronized on the Publisher channel and not on the Subscriber channel. This is valid behavior when synchronizing on the Publisher channel. ◆ **None:** No merging occurs regardless of synchronization.
Optimize Modification to Identity Vault	◆ **Yes:** Changes to this attribute are examined on the Publisher channel to determine the minimal change made in the Identity Vault. ◆ **No:** Changes are not examined.

5 Click *OK* to save the changes.

Using Predefined Rules

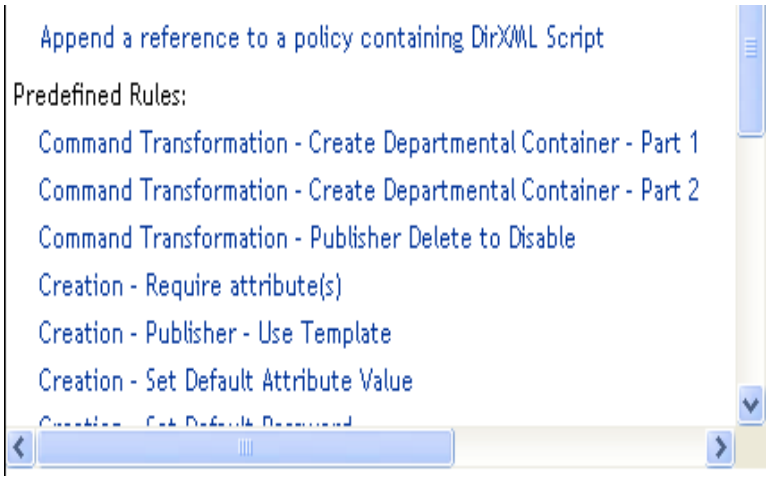
6

iManager 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. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see “Accessing the Identity Manager Driver Overview Page” on page 270.
2. Click the icon representing the policy where you want to add the predefined rule.
3. Click the name of the policy.
4. Click *Insert* and select the predefined rule you want to use.



6.1 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.

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 48.

6.1.1 Creating a Policy

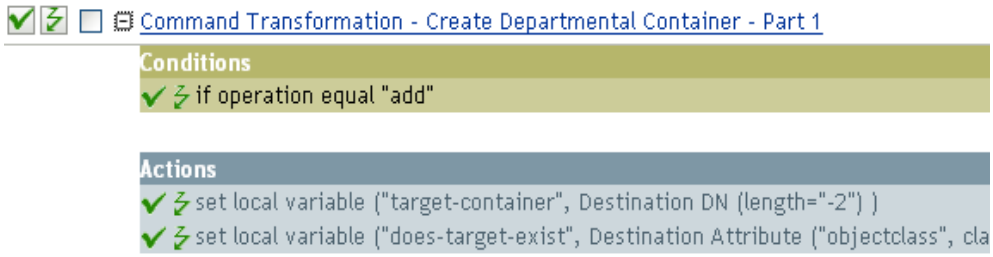
1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Command Transformation Policy set object on the Publisher or Subscriber channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Policy Builder is launched.
5. Continue with Section 6.1.2, "Importing the Predefined Rule," on page 48.

6.1.2 Importing the Predefined Rule

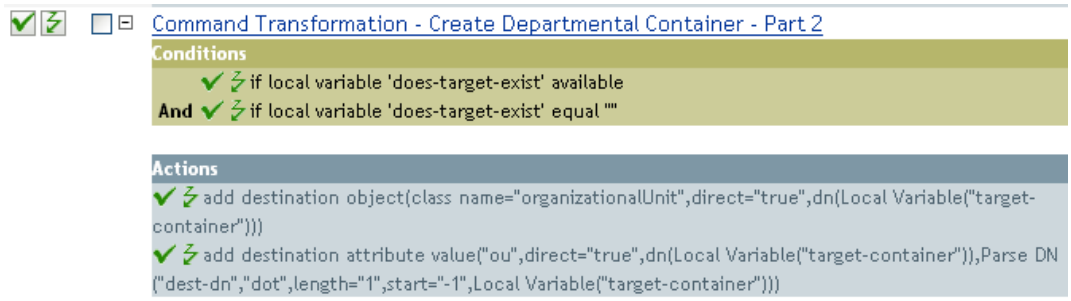
1. In the Policy Builder, click *Insert*.
2. Select *Command Transformation - Create Departmental Container - Part 1*.
3. Expand the predefined rule.



4 Click *Insert*.

5 Select *Command Transformation - Create Departmental Container - Part 2*.

6 Expand the predefined rule.



7 Click *OK*.

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

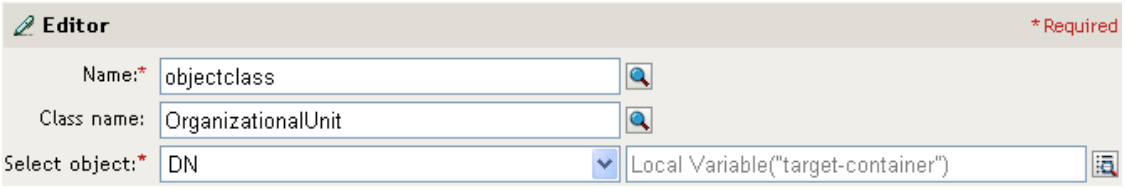
IMPORTANT: Make sure that the rules are listed in order. Part 1 must be executed before Part 2.

6.1.3 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 operation. When the Add operation 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.

Figure 6-1 *Create 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 name of the new organizational unit, which is obtained by parsing the value of the local variable target-container.

For more information on the Editor and how to access it, see "Argument Builder" on page 26.

6.2 Command Transformation - Publisher Delete to Disable

This rule transforms a Delete operation for a User object into a Modify operation that disables the target User object in eDirectory™. Implement the rule on the Publisher Command Transformation policy in the driver.

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 (page 50).

6.2.1 Creating a Policy

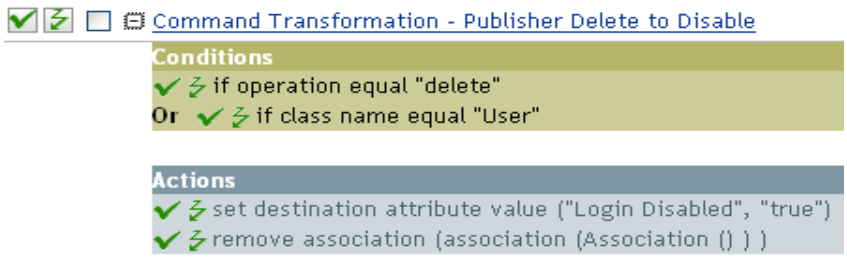
1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Command Transformation Policy set object on the Publisher channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Policy Builder is launched.
5. Continue with Section 6.2.2, "Importing the Predefined Rule," on page 50.

6.2.2 Importing the Predefined Rule

1. In the Policy Builder, click *Insert*.
2. Select *Command Transformation - Publisher Delete to Disable*.
3. Expand the predefined rule.



4. Click OK.

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

6.2.3 How the Rule Works

This rule is used when a Delete command is going to be sent to the Identity Vault, usually in response to a Delete event that occurred in the connected system. Instead of the User object being deleted in the Identity Vault, the User object is disabled. When a Delete command is processed for a User object, the destination attribute value of Login Disabled is set to true, the association is removed from the User object, and the Delete command is vetoed. The User object can no longer log in into the eDirectory tree, but the User object was not deleted.

6.3 Creation - Require Attributes

This rule prevents User objects from being created unless the required attributes are populated. Implement the rule on the Subscriber Creation policy or the 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 51.

6.3.1 Creating a Policy

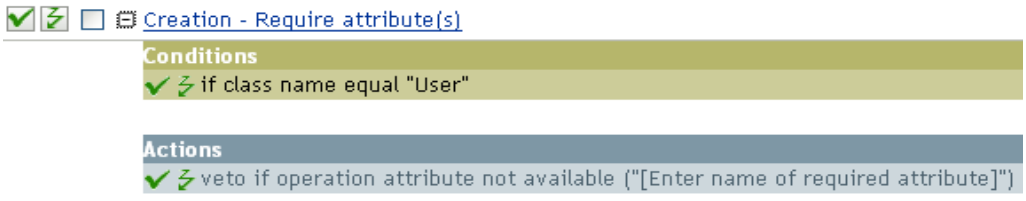
1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Creation Policy set object on the Publisher or Subscriber channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Policy Builder is launched.
5. Continue with Section 6.3.2, "Importing the Predefined Rule," on page 51.

6.3.2 Importing the Predefined Rule

1. In the Policy Builder, click *Insert*.
2. Select *Creation - Required Attributes*.
3. Expand the predefined rule.



4. To edit the rule, click *Creation - Required Attributes* in the Policy Builder.

   The Rule Builder is launched.
5. In the *Conditions* section, click the *Browse* icon next to the *Value* field.
6. Browse to and select the attribute you require for a User object to be created.

7 (Optional) If you want more than one required attribute, click the plus icon to add a new action.

8 Select *Veto if operation attribute not available*, then browse to and select the additional required attribute.

9 Click *OK* twice.

6.3.3 How the Rule Works

This rule is used when your business processes require that a user has specific attributes populated in the source User object before the destination User object can be created. When a User object is created in the source data store, the rule vetoes the creation of the object in the destination data store unless the required attributes are provided when the User object is created. You can have one or more required attributes.

6.4 Creation - Publisher - Use Template

This rule allows for 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.

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 52.

6.4.1 Creating a Policy

1 Open the Identity Manager Driver Overview for the driver you want to manage.

For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click the Creation Policy set object on the Publisher or Subscriber channel.

3 Click *Insert*.

4 Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

The Policy Builder is launched.

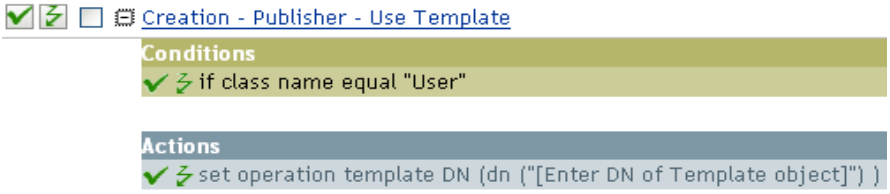
5 Continue with Section 6.4.2, "Importing the Predefined Rule," on page 52.

6.4.2 Importing the Predefined Rule

1 In the Policy Builder, click *Insert*.

2 Select *Creation - Publisher - Use Template*.

3 Expand the predefined rule.



4 To edit the rule, click *Creation - Publisher - Use Template* in the Policy Builder.

The Rule Builder is launched.

5 In the *Actions* section, click the *Edit the arguments* icon.

The Argument Builder is launched.

6 In the Editor, click the *Browse* icon next to the *Text* field, browse to and select the template object, then click *OK*.

7 Click *OK*.

6.4.3 How the Rule Works

This rule is used when you want to create a user in the Identity Vault based on a template object. If you have attributes that are the same for users, using the template saves time. You fill in the information in the template object. When the User object is created, Identity Manager uses the attribute values from the template to create the User object.

During the creation of User objects, the rule does the action of the set operation template DN, which instructs the Identity Manager to use the referenced template when creating the object.

6.5 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 53.

6.5.1 Creating a Policy

1 Open the Identity Manager Driver Overview for the driver you want to manage.

For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click the Creation Policy object on the Publisher or Subscriber channel.

3 Click *Insert*.

4 Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

The Policy Builder is launched.

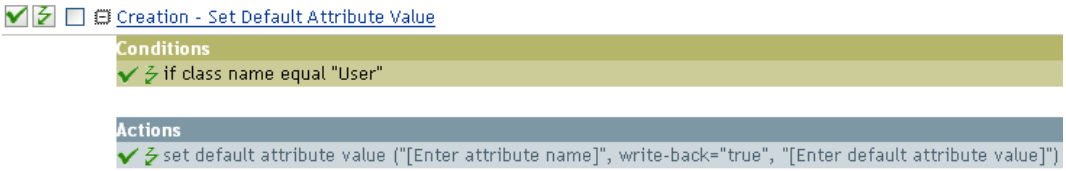
5 Continue with Section 6.5.2, "Importing the Predefined Rule," on page 53.

6.5.2 Importing the Predefined Rule

1 In the Policy Builder, click *Insert*.

2 Select *Creation - Set Default Attribute Value*.

3 Expand the predefined rule.



4 To edit the rule, click *Creation - Set Default Attribute Value* in the Policy Builder.

   The Rule Builder is launched.

5 In the *Action* section, click the *Browse* icon next to the *Enter attribute name* field, then browse to and select the attribute you want to have created.

6 Click the *Edit the value list* icon next to the *Enter argument values* field.

   The Argument Value List Builder is launched.

7 Browse to and select the type of data you want the value to be.

8 Click the *Edit the arguments* icon.

   The Argument Builder is launched.

9 Delete *[Edit default attribute value]* from the Argument Builder by selecting it and clicking the *Remove the selected token* icon.

10 In the Editor, click the browse button next to the *Text* field, then browse to and select the container in the desination hierarchy where you want the source

11 Click *OK*.

6.5.3 How the Rule Works

This rule is used when you want to populate default attribute values when creating a User object. When a User object is created, the rule adds the specified attribute values if and only if the attribute has no values supplied by the source object.

If you want more than one attribute value defined, right-click the action and click *New* > *Action*. Select the action, set the default attribute value, and follow the steps above to assign the value to the attribute.

6.6 Creation - Set Default Password

During the creation of User objects, this rule sets a default password for User objects. Implement the rule on the 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 55.

6.6.1 Creating a Policy

1 Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click the Creation Policy object on the Publisher or Subscriber channel.

3 Click *Insert*.

4 Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

The Policy Builder is launched.

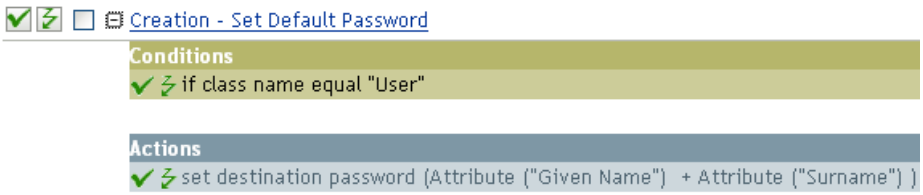
5 Continue with Section 6.6.2, "Importing the Predefined Rule," on page 55.

6.6.2 Importing the Predefined Rule

1 In the Policy Builder, click *Insert*.

2 Select *Creation - Set Default Password*.

3 Expand the predefined rule.



4 Click *OK*.

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

6.6.3 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.

6.7 Event Transformation - Scope Filtering - Include Subtrees

This rule excludes all events that occur outside of the specific subtrees. Implement the rule on the Subscriber Event Transformation policy or the Publisher Event Transformation policy in the driver.

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 56.

6.7.1 Creating a Policy

1 Open the Identity Manager Driver Overview for the driver you want to manage.

For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click the Event Transformation Policy set object on the Publisher or Subscriber channel.

3 Click *Insert*.

4 Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

The Policy Builder is launched.

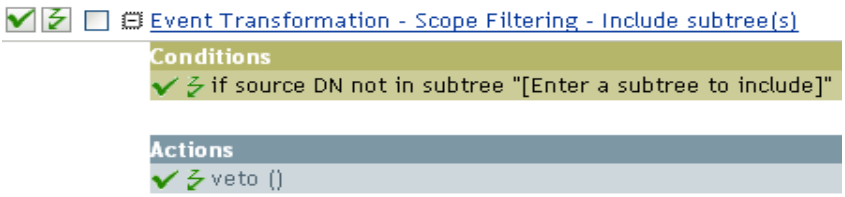
5 Continue with Section 6.7.2, "Importing the Predefined Rule," on page 56.

6.7.2 Importing the Predefined Rule

1 In the Policy Builder, click *Insert*.

2 Select *Event Transformation - Scope Filtering - Include subtrees*.

3 Expand the predefined rule.



4 To edit the rule, click *Event Transformation - Scope Filtering - Include subtrees* in the Policy Builder.

The Rule Builder is launched.

5 Click the browse button next to the *Value* field to browse the Identity Vault for the part of the tree where you want events to synchronize, select it, then click *OK*.

6 Click *OK*.

6.7.3 How the Rule Works

This rule is used when you only want to synchronize specific subtrees between the Identity vault and the connected system.When an event occurs anywhere but in that specific part of the Identity Vault, it is vetoed. You can add additional subtrees to be synchronized by copying and pasting the If Source DN condition.

6.8 Event Transformation - Scope Filtering - Exclude Subtrees

This rule excludes all events that occur in a specific subtree. Implement the rule on the Subscriber Event Transformation or the Publisher Event Transformation policy in the driver.

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 57.

6.8.1 Creating a Policy

1 Open the Identity Manager Driver Overview for the driver you want to manage.

For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click the Event Transformation Policies set object on the Publisher or Subscriber channel.

3 Click *Insert*.

4 Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

The Policy Builder is launched.

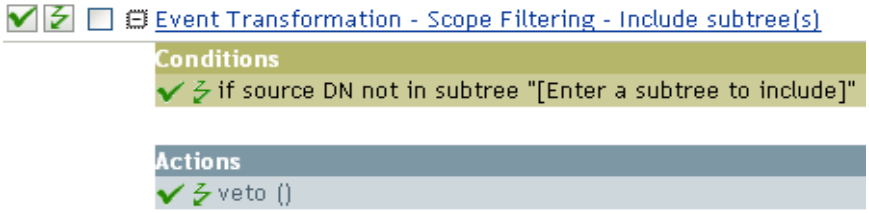
5 Continue with Section 6.8.2, "Importing the Predefined Rule," on page 57.

6.8.2 Importing the Predefined Rule

1 In the Policy Builder, click *Insert*.

2 Select *Event Transformation - Scope Filtering - Exclude subtrees*.

3 Expand the predefined rule.



4 To edit the rule, click *Event Transformation - Scope Filtering - Exclude subtrees* in the Policy Builder.

The Rule Builder is launched.

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

6 Click *OK*.

6.8.3 How the Rule Works

This rule is used when you want to exclude part of the Identity Vault or connected system from synchronizing. When an event occurs in that specific part of the Identity Vault, it is vetoed. You can add additional subtrees to be excluded by copying and pasting the If Source DN condition.

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

This rule converts the format of the telephone number. Implement the rule on the Input or Output Transformation policy in the driver. Typically, if this rule is used on an Input Transformation, you would then use the rule Reformat Telephone Number from nnn-nnn-nnnn to (nnn) nnn-nnnn on the Output Transformation and vice versa to convert the format back and forth.

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 58.

6.9.1 Creating a Policy

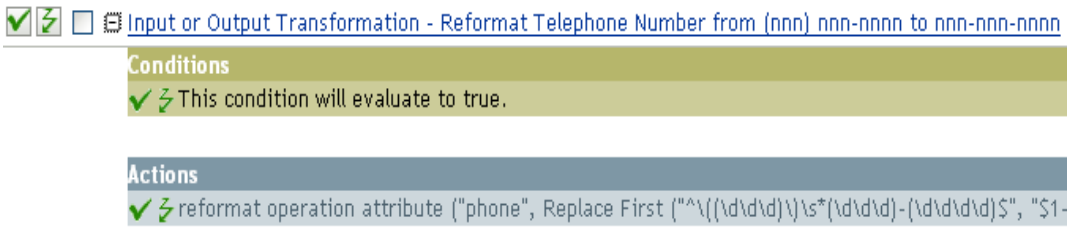
1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Input or Output Transformation Policy set object on the Publisher or Subscriber channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Policy Builder is launched.
5. Continue with Section 6.9.2, "Importing the Predefined Rule," on page 58.

6.9.2 Importing the Predefined Rule

1. In the Policy Builder, click *Insert*.
2. Select *Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn*.
3. Expand the predefined rule.



4. To edit the rule, click *Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn* in the Policy Builder.

   The Rule Builder is launched.
5. Define the condition you want to have occur when the telephone number is reformatted.
6. Click *OK*.

6.9.3 How the Rule Works

This rule is used when you want to reformat the telephone number. It finds all the values for the phone attribute in the current operation that match the pattern (nnn) nnn-nnnn and replaces each with nnn-nnn-nnnn.

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

This rule transforms the format of the telephone number. Implement the rule on the Input or Output Transformation policy. Typically, if you use this rule on an Output Transformation, you would use the rule Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn on the Input Transformation and vice versa to convert the format back and forth.

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 59.

6.10.1 Creating a Policy

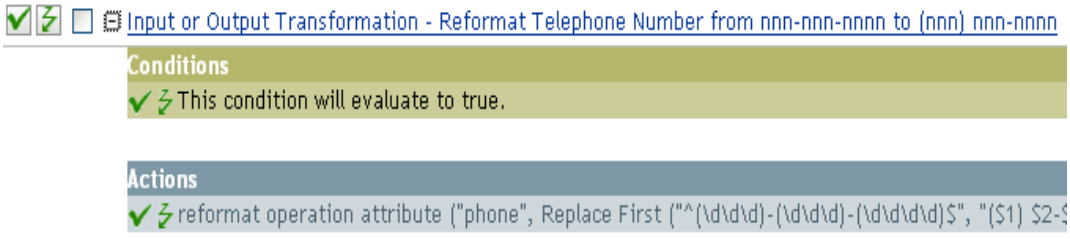
1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Input or Output Transformation Policy set object on the Publisher or Subscriber channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Policy Builder is launched.
5. Continue with Section 6.10.2, "Importing the Predefined Rule," on page 59.

6.10.2 Importing the Predefined Rule

1. In the Policy Builder, click *Insert*.
2. Select *Input or Output Transformation - Reformat Telephone Number from nnn-nnn-nnnn to (nnn) nnn-nnnn*.
3. Expand the predefined rule.



4. To edit the rule, click *Input or Output Transformation - Reformat Telephone Number from nnn-nnn-nnnn to (nnn) nnn-nnnn* in the Policy Builder.

   The Rule Builder is launched.
5. Define the condition you want to have occur when the telephone number is reformatted.
6. Click *OK*.

6.10.3 How the Rule Works

This rule is used when you want to reformat the telephone number. It finds all the values for the phone attribute in the current operation that match the pattern (nnn) nnn-nnnn and replaces each with nnn-nnn-nnnn.

6.11 Matching - Publisher Mirrored

This rule finds matches in the Identity Vault for objects in the connected system based on their name and location. Implement the rule on the Publisher Matching policy in the driver.

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 60.

6.11.1 Creating a Policy

1. Open the Identity Manager Driver Overview for the driver you want to manage.

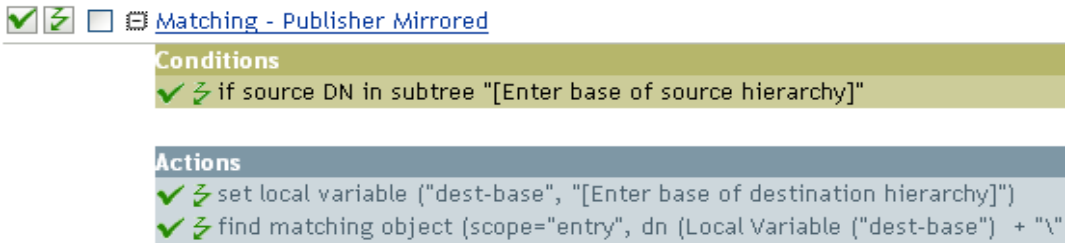
   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Matching Policy set object on the Publisher channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Policy Builder is launched.
5. Continue with Section 6.11.2, "Importing the Predefined Rule," on page 60.

6.11.2 Importing the Predefined Rule

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

   For information on how to access the policy builder, see "Accessing the Policy Builder" on page 15.
2. Select *Matching - Publisher Mirrored*.
3. Expand the predefined rule.



4. To edit the rule, click *Matching - Publisher Mirrored* in the Policy Builder.

   The Rule Builder is launched.
5. In the *Conditions* section, click the *Browse* icon next to the *Value* field.
6. Click the container in the source hierarchy where you want the matching to start.

7 In the *Actions* section, click the *Edit the arguments* icon next to the *Enter string* field.

8 In the Editor, click the browse button next to the *Text* field, browse to and select the container in the destination hierarchy where you want the source structure to be matched, then click *OK*.

9 Click *OK*.

6.11.3 How the Rule Works

When an Add event occurs on an object in the connected system that is located within the specified source subtree, the rule constructs a DN that represents the same object name and location within the Identity Vault relative to the specified destination subtree. If the destination objects exists and is of the desired object class, then it is considered a match. You must supply the DNs of the source (connected system) and destination (Identity Vault) subtrees.

6.12 Matching - Subscriber Mirrored - LDAP Format

This rule finds matches in a connected system that uses LDAP format DNs for objects in the Identity Vault based on their names and locations. Implement the rule on the Subscriber Matching policy in the driver.

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 61.

6.12.1 Creating a Policy

1 Open the Identity Manager Driver Overview for the driver you want to manage.

For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click the Matching Policy set object on the Subscriber channel.

3 Click *Insert*.

4 Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

The Policy Builder is launched.

5 Continue with Section 6.12.2, "Importing the Predefined Rule," on page 61.

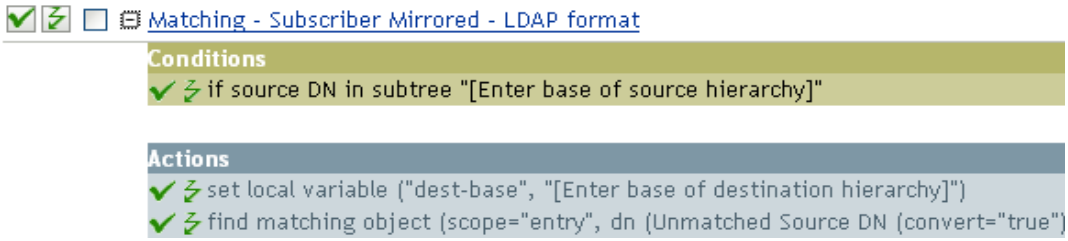
6.12.2 Importing the Predefined Rule

1 In the Policy Builder, click *Insert*.

For information on how to access the policy builder, see "Accessing the Policy Builder" on page 15.

2 Select *Matching - Subscriber Mirrored - LDAP format*.

3 Expand the predefined rule.



4 To edit the rule, click *Matching - Subscriber Mirrored - LDAP format* in the Policy Builder.

The Rule Builder is launched.

5 In the *Condition* section, click the *Browse* icon next to the *Value* field.

6 Click the container in the source hierarchy where you want the matching to start.

7 In the *Actions* section, click the *Edit the arguments* icon next to the *Enter string* field.

8 In the Editor, click the browse button next to the *Text* field, browse to and select the container in the destination hierarchy where you want the source structure to be matched, then click *OK*.

9 Click *OK*.

6.12.3 How the Rule Works

When an Add event occurs on an object in the Identity Vault that is located within the specified source subtree, the rule constructs a DN that represents the same object name and location within the connected system relative to the specified destination subtree. If the destination objects exists and is of the desired object class, then it is considered a match. You must supply the DNs of the source (Identity Vault) and destination (connected system) subtrees. The connected system must use an LDAP-formatted DN.

6.13 Matching - By Attribute Value

This rule finds matches for objects by specific attribute values. Implement the rule on the Subscriber Matching policy or the Publisher Matching policy in the driver.

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 63.

6.13.1 Creating a Policy

1 Open the Identity Manager Driver Overview for the driver you want to manage.

For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click the Matching Policies set object on the Publisher or Subscriber channel.

3 Click *Insert*.

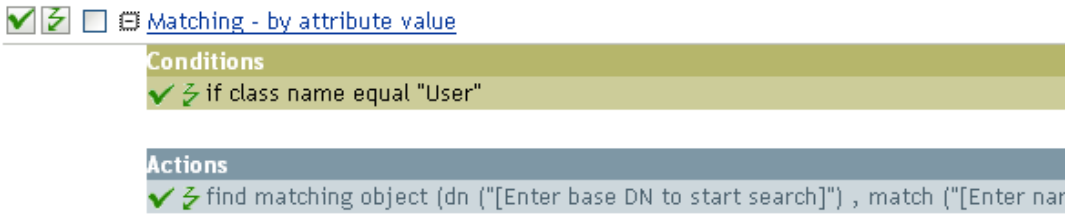
4 Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

The Policy Builder is launched.

5 Continue with Section 6.13.2, "Importing the Predefined Rule," on page 63.

6.13.2 Importing the Predefined Rule

1. In the Policy Builder, click *Insert*.
2. Select *Matching - By Attribute Value*.
3. Expand the predefined rule.



4. To edit the rule, click *Matching - By Attribute Value* in the Policy Builder.

   The Rule Builder is launched.
5. Click the *Edit the arguments* icon by the *Enter DN* field to launch the Argument Builder.
6. In the Editor, click the browse button, browse to and select the container where you want the search to start, then click *OK*.
7. In the *Action* section, click the *Edit the match attributes* icon to launch the Match Attribute Builder.
8. Click the browse button next to the *Name* field and select the attributes you want to match. You can select one or more attributes to match against. Click *OK*.
9. Click *OK*.

6.13.3 How the Rule Works

When an Add event occurs on an object in the source data store, the rule searches for an object in the destination data store that has the same values for the specified attribute. You must supply the DN of the base of the subtree to search in the connected system and the name of the attribute to match on.

6.14 Placement - Publisher Mirrored

This rule places objects in the Identity Vault based on the name and location from the connected system. Implement the rule on the Publisher 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 64.

6.14.1 Creating a Policy

1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Placement Policies set object on the Publisher channel.
3. Click *Insert*.

4 Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

The Policy Builder is launched.

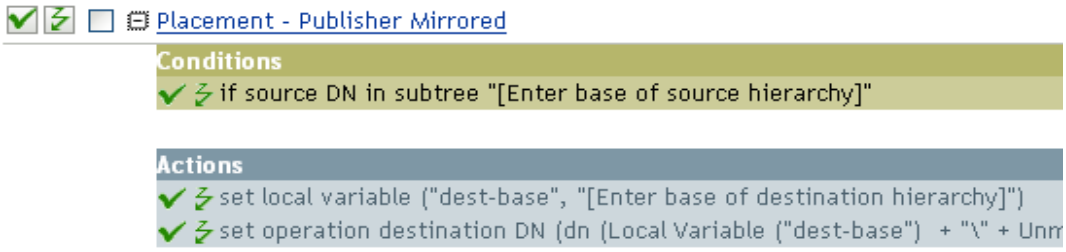
5 Continue with Section 6.14.2, “Importing the Predefined Rule,” on page 64.

6.14.2 Importing the Predefined Rule

1 In the Policy Builder, click *Insert*.

2 Select *Placement - Publisher Mirrored*.

3 Expand the predefined rule.



4 To edit the rule, click *Placement - Publisher Mirrored* in the Policy Builder.

The Rule Builder is launched.

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

6 Click the *Edit the arguments* icon next to the *Enter string* field.

The Argument Builder is launched.

7 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*.

8 Click *OK*.

6.14.3 How the Rule Works

If the User object resides in the specified source subtree in the connected system, then the object is placed at the same relative name and location within the Identity Vault. You must supply the DNs of the source (connected system) and destination (Identity Vault) subtrees.

6.15 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 65.

6.15.1 Creating a Policy

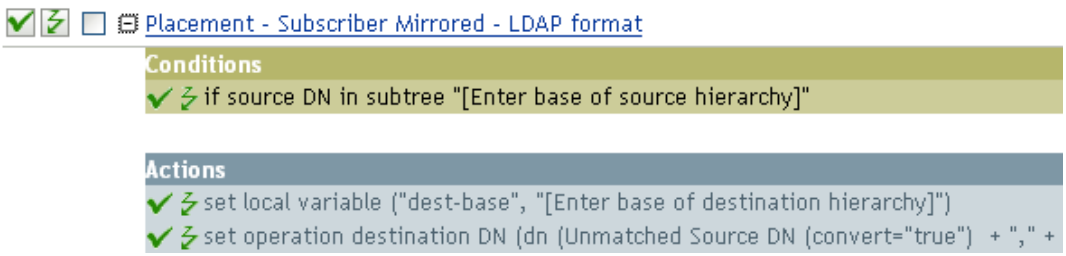
1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Placement Policies set object on the Subscriber channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Policy Builder is launched.
5. Continue with Section 6.15.2, "Importing the Predefined Rule," on page 65.

6.15.2 Importing the Predefined Rule

1. In the Policy Builder, click *Insert*.
2. Select *Placement - Subscriber Mirrored - LDAP Format*.
3. Expand the predefined rule.



4. To edit the rule, click *Placement - Subscriber Mirrored - LDAP Format* in the Policy Builder.

   The Rule Builder is launched.
5. In the *Value* field, browse to and click the container in the source hierarchy where you want the object to be acted upon.
6. Click the *Edit the arguments* icon next to the *Enter string* field.

   The Argument Builder is launched.
7. 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*.
8. Click *OK*.

6.15.3 How the Rule Works

If the User object resides in the specified source subtree, the object is placed at the same relative name and location within the Identity Vault. You must supply the DNs of the source (Identity Vault) and destination (connected system) subtrees. The connected system must use an LDAP-formatted DN.

6.16 Placement - Publisher Flat

This rule places objects from the data store into one container in the Identity Vault. Implement the rule on the Publisher 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 66.

6.16.1 Creating a Policy

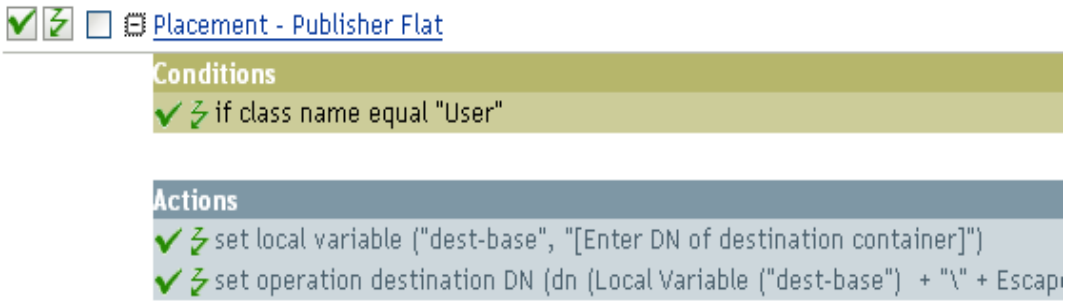
1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Placement Policies set object on the Publisher channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Policy Builder is launched.
5. Continue with Section 6.16.2, "Importing the Predefined Rule," on page 66.

6.16.2 Importing the Predefined Rule

1. In the Policy Builder, click *Insert*.
2. Select *Placement - Publisher Flat*.
3. Expand the predefined rule.



4. To edit the rule, click *Placement - Publisher Flat* in the Policy Builder.

   The Rule Builder is launched.
5. In the *Enter string* field, click the *Edit the arguments* icon.

   The Argument Builder is launched.
6. In the Editor, click the browse button, browse to and select the destination container were you want all of the user objects to be placed, then click *OK*.
7. Click *OK*.

6.16.3 How the Rule Works

The 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 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.

6.17 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 67.

6.17.1 Creating a Policy

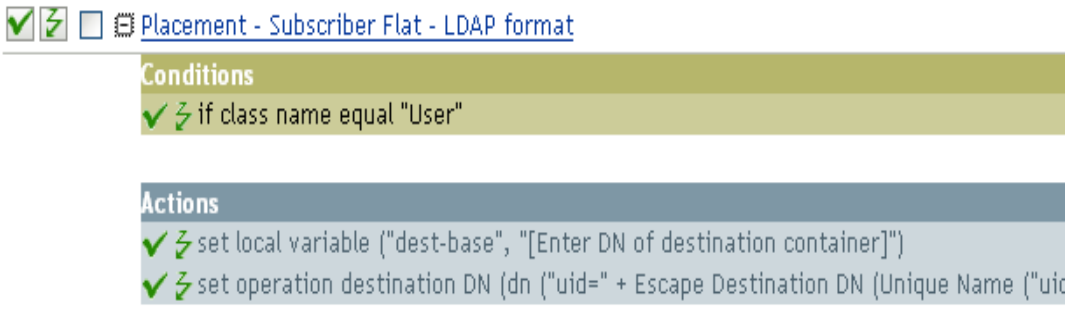
1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Placement Policies set object on the Subscriber channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Rule Builder is launched.
5. Continue with Section 6.17.2, "Importing the Predefined Rule," on page 67.

6.17.2 Importing the Predefined Rule

1. In the Rule Builder, click *Insert*.
2. Select *Placement - Subscriber Flat - LDAP Format*.
3. Expand the predefined rule.



4. To edit the rule, click *Placement - Subscriber Flat - LDAP Format* in the Policy Builder.

   The Rule Builder is launched.
5. In the *Enter string* field, click the *Edit the arguments* icon.

The Argument Builder is launched.

6 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*.

7 Click *OK*.

6.17.3 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 as lowercase. The rule uses LDAP format.

6.18 Placement - Publisher By Dept

This rule places objects from one container in the data store into multiple containers in the Identity Vault based on the value of the OU attribute. Implement the rule on the Publisher 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 68.

6.18.1 Creating a Policy

1 Open the Identity Manager Driver Overview for the driver you want to manage.

For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click the Placement Policies set object on the Publisher channel.

3 Click *Insert*.

4 Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

The Policy Builder is launched.

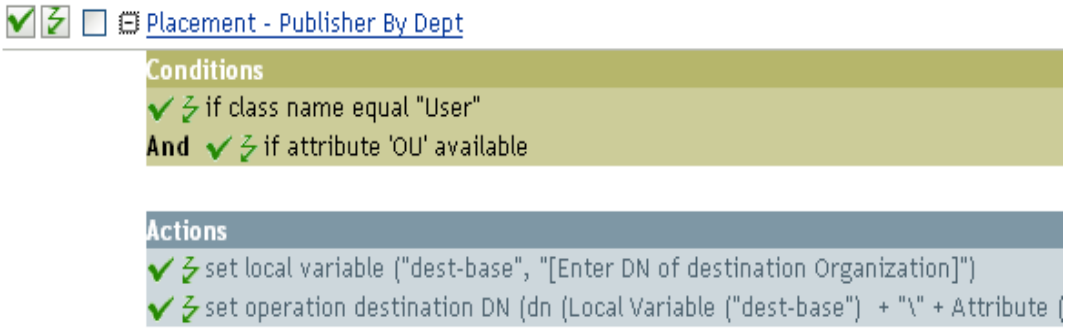
5 Continue with Section 6.18.2, "Importing the Predefined Rule," on page 68.

6.18.2 Importing the Predefined Rule

1 In the Policy Builder, click *Insert*.

2 Select *Placement - Publisher By Dept*.

3 Expand the predefined rule.



4 To edit the rule, click *Placement - Publisher By Dept* in the Policy Builder.

   The Rule Builder is launched.

5 In the *Enter string* field, click the *Edit the arguments* icon.

   The Argument Builder is launched.

6 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*.

7 Click *OK*.

6.18.3 How the Rule Works

This rule places User objects in the correct department containers depending upon what value 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 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 as lowercase. The rule uses slash format.

6.19 Placement - Subscriber By Dept - LDAP Format

This rule places objects from one container in the Identity Vault into multiple containers in the data store 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 70.

6.19.1 Creating a Policy

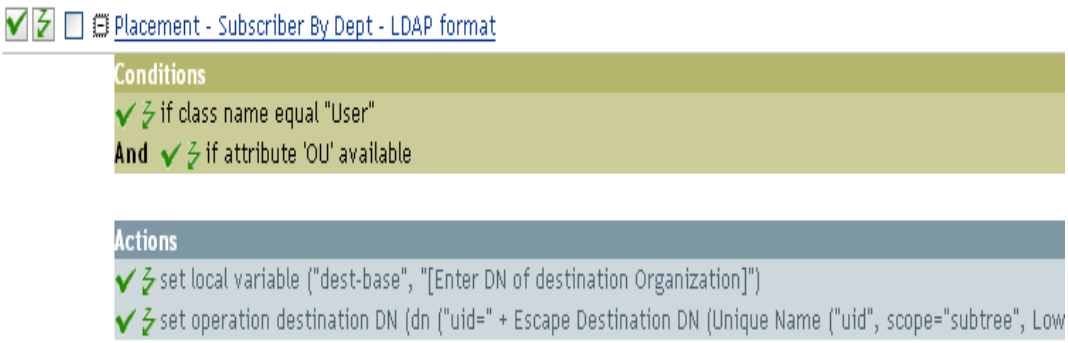
1. Open the Identity Manager Driver Overview for the driver you want to manage.

   For instructions on how to access the Identity Manager Driver Overview page, see "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click the Placement Policies set object on the Subscriber channel.
3. Click *Insert*.
4. Name the policy, make sure to implement the policy with the Policy Builder, then click *OK*.

   The Policy Builder is launched.
5. Continue with Section 6.19.2, "Importing the Predefined Rule," on page 70.

6.19.2 Importing the Predefined Rule

1. In the Policy Builder, click *Insert*.
2. Select *Placement - Subscriber By Dept - LDAP format*.
3. Expand the predefined rule.



4. To edit the rule, click *Placement - Subscriber By Dept - LDAP format* in the Policy Builder.

   The Rule Builder is launched.
5. In the *Enter string* field, click the *Edit the arguments* icon.

   The Argument Builder is launched.
6. 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*.
7. Click *OK*.

6.19.3 How the Rule Works

This rule places User objects in the correct department containers depending upon what value 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 place 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 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.

Storing Information in Resource Objects

7

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

7.1 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 that is running the driver holds a Read/Write or Master replica of the library object.

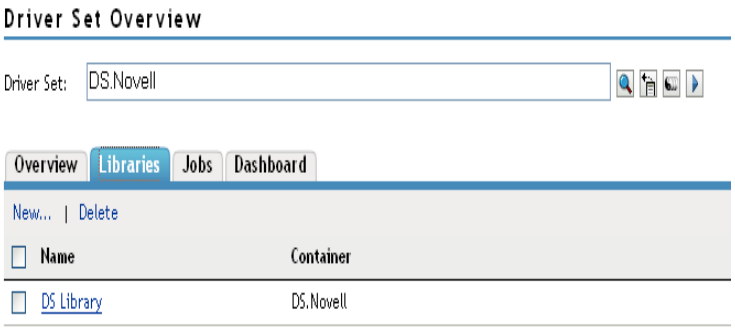
Style sheets, policies, rules, and other resource objects can be stored in a library and be referenced by one or more drivers.

7.1.1 Managing Libraries

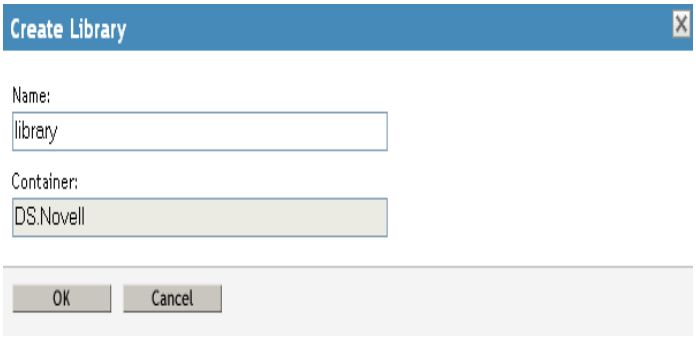
You can create, delete, and search for existing libraries in iManager.

Creating a Library

1. Access the Identity Manager Driver Set Overview page by following the steps in “Accessing the Identity Manager Driver Set Overview Page” on page 269.
2. Click the *Libraries* tab.



3 Click *New*.



4 Specify a name for the library.

5 The library is created in the container that was previously selected.

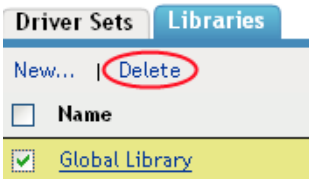
6 Click *OK*.

Deleting a Library

1 Access the Identity Manager Driver Set Overview page by following the steps in "Accessing the Identity Manager Driver Set Overview Page" on page 269.

2 Click the *Libraries* tab.

3 Select the library you want to delete, then click *Delete*.



4 Click *OK* to confirm the deletion.

7.1.2 Adding Objects to the Library

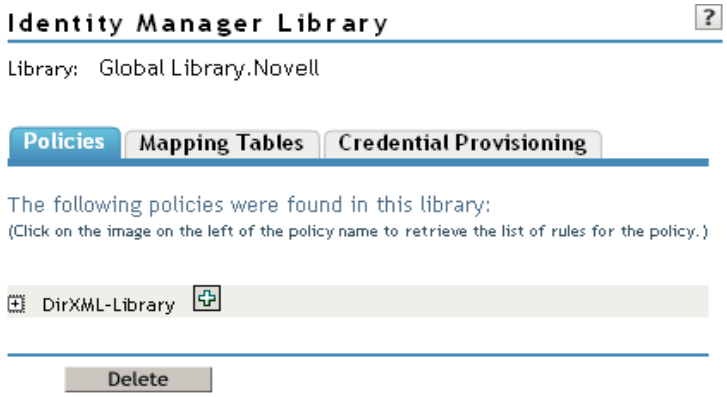
You can add policies, mapping tables, and Credential Provisioning policy resource objects to a library.

- “Adding Policies to the Library” on page 75
- “Adding a Mapping Table to a Library” on page 76
- “Adding Credential Provisioning Policy Resource Objects to a Library” on page 77

Adding Policies to the Library

1. Access the Identity Manager Driver Set Overview page by following the steps in “Accessing the Identity Manager Driver Set Overview Page” on page 269.
2. Click the *Libraries* tab.
3. Click the library you want to add a policy to.

4. Click the *Policies* tab, then click the plus icon to add a policy to the library.

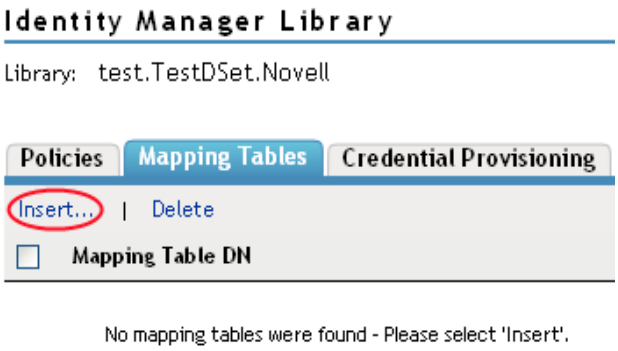


5. Specify the name for the policy.
6. Select how to implement the policy, then click *OK*.
   - If you select *Policy Builder*, *XSLT*, or *ECMAScript*, the object is created and displayed in the library. Each object must be edited to add the policy information into the object.
   - If you select *Make a copy from an existing policy*, browse to and select the policy to store in the library.

Adding a Mapping Table to a Library

1. Access the Identity Manager Driver Set Overview page by following the steps in "Accessing the Identity Manager Driver Set Overview Page" on page 269.
2. Click the *Libraries* tab.
3. Click the library you want to add a mapping table to.

4. Click the *Mapping Tables* tab, then click *Insert* to add a mapping table to the library.



5. Specify the name for the mapping table.
6. Browse to and select the library where the mapping table will be created.
7. Click *OK*.

   The Mapping Table Editor is launched.
8. Click the *Add a column to the mapping table* icon.
9. Specify a value for the column, then select whether the value is case sensitive, case insensitive, or numeric.
10. Click the *Add Row* icon.
11. Specify a value for the row.
12. Click *Apply* to save the mapping table and continue working in the editor

    or

    Click *OK* to save the mapping table and close the editor.

For more information about mapping tables, see Section 7.2, "Mapping Table Objects," on page 78.

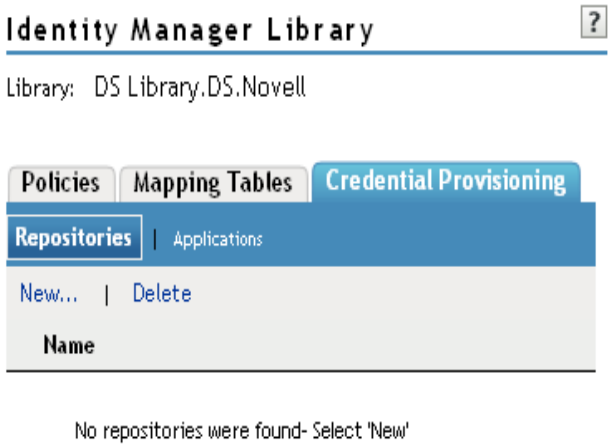
Adding Credential Provisioning Policy Resource Objects to a Library

1. Access the Identity Manager Driver Set Overview page by following the steps in "Accessing the Identity Manager Driver Set Overview Page" on page 269.
2. Click the *Libraries* tab.
3. Click the library you want to add a Credential Provisioning policy resource object to.

4. Click the *Credential Provisioning* tab.
5. Click *Repositories*, then click *New* to add a new repository object to the library.

   or

   Click *Applications*, then click *New* to add a new application object to the library.



6. Click *OK*.

7.1.3 Using a Policy Stored in the Library

The library object stores information that is used multiple times. It can be used by multiple drivers or by the same driver multiple times. To use the policy stored in the library:

1. Access the Identity Manager Driver Overview page by following the steps in "Accessing the Identity Manager Driver Overview Page" on page 270.
2. Click a policy set, click *Insert*, then proceed to Step 3.

   or

   Click an existing policy, then skip to Step 6.
3. Select *Use an existing policy*.

4 Browse to and select the policy that is stored in the library, then click *OK*.

5 Click *Close*.

6 Click *Insert* > *Append a reference to a policy containing DirXML Script*.

7 Browse to and select the policy that is stored in the library, then click *OK* twice.

8 Click *Close*.

7.2 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 255) 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:

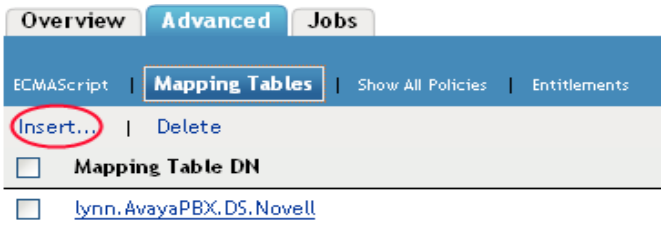
1. Section 7.2.1, "Creating a Mapping Table Object," on page 78
2. Section 7.2.2, "Adding a Mapping Table Object to a Policy," on page 79

7.2.1 Creating a Mapping Table Object

1 Access the Identity Manager Driver Overview page, by following the steps in "Accessing the Identity Manager Driver Overview Page" on page 270.

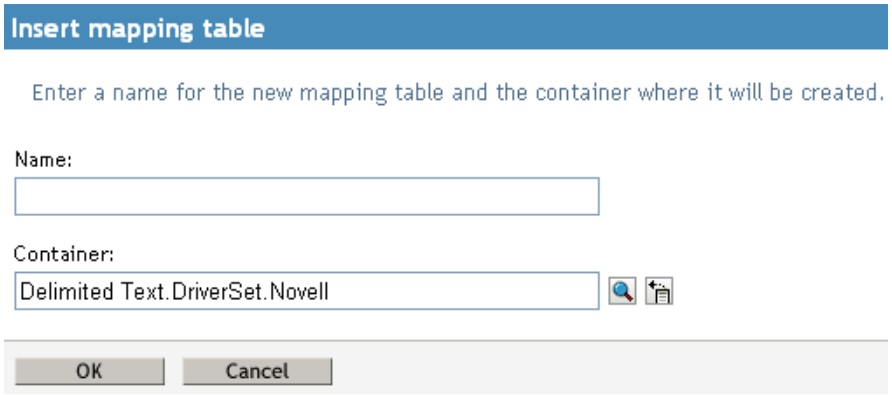
Choose the driver where you want to create the mapping table.

2 Select *Advanced* > *Mapping Tables*, then click *Insert*.

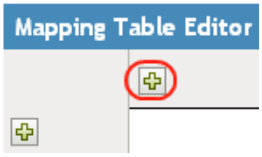


3 Specify the name of the mapping table object.

4 Browse to and select the container where the mapping table will be created, then click *OK*.



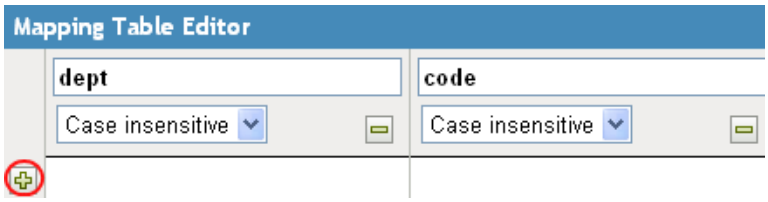
5 Click the *Add Column* icon.



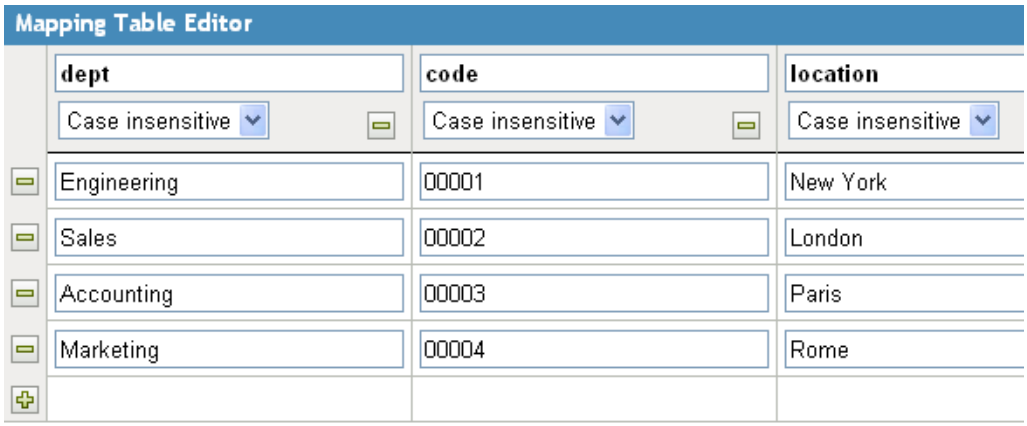
6 Specify the name of the column, then select whether the value is *Case insensitive*, *Case sensitive*, or *Numeric*.

If you want to add more columns, repeat Step 5 and Step 6.

7 Click the *Add Row* icon.



8 Specify the value for the row.



If you want more rows, repeat Step 7 and Step 8.

9 Click *OK* to save the mapping table and exit the Mapping Table editor.

7.2.2 Adding a Mapping Table Object to a Policy

1 Access the Identity Manager Driver Overview page by following the steps in "Accessing the Identity Manager Driver Overview Page" on page 270.

2 Click a policy set where you want to add a mapping table object.

3 Create a policy to use the mapping table in. For instructions on how to do this, see "Creating a Policy in a Driver" on page 15.

or

Click an existing policy to edit.

The Policy Builder is displayed.

4 If you created your own policy, or if there are no rules for your policy, create a rule for the policy. For information on how to do this, see "Defining Individual Rules within a Policy" on page 18.

5 Click a rule.

The Rule Builder is launched.

6 Create a rule that contains an action that would call the mapping table.

7 Launch the Argument Builder in the Rule Builder by clicking the *Edit the arguments* icon in the *Action List* section.

8 Select *Map* from the list of *Verbs*, then click *Add*.

9 In the *Editor* field, browse to and select the mapping table object created in Section 7.2.1, "Creating a Mapping Table Object," on page 78.

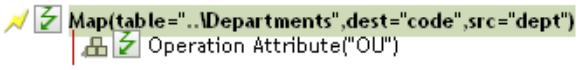
10 Specify the source column name.

11 Specify the destination column name.

12 (Optional) Define the default value for the destination column.

13 Select a Noun to achieve the desired results, then click *OK* to save the argument.

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.



The Map token is a Verb token. It requires a Noun token to act upon in order to function.

7.3 ECMAScript

ECMAScript objects are resource objects that store ECMAScripts, which are used by policies and style sheets. For more information on ECMAScript, see Chapter 8, "Using ECMAScript in Policies," on page 83.

7.4 Application Objects

Application objects are part of Novell® Credential Provisioning policies. The application objects store application authentication parameter values for SecureLogin. For information about application objects, see *Novell Credential Provisioning Policies for Identity Manager 3.6*.

7.5 Repository Objects

Repository objects are part of Novell Credential Provisioning policies. The repository objects store static configuration information for SecureLogin. For information about repository objects, see *Novell Credential Provisioning Policies for Identity Manager 3.6*.

7.6 Resource Objects

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.

At this time, the supported way to create resource objects is through Designer. For more information, see "Storing Information in Resource Objects" in *Policies in Designer 3.0*.

Using ECMAScript in Policies

8

ECMAScript is a scripting programming language, standardized by Ecma International. It is often referred to as JavaScript* or JScript*, but these are subsets of ECMAScript. Identity Manager 3.5.1 and later 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.

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.

8.1 Creating an ECMAScript

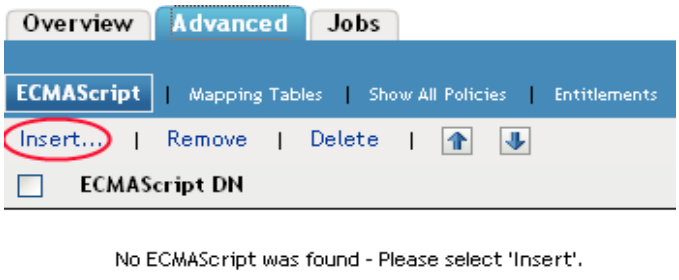
An ECMAScript is stored on a driver or in a library.

8.1.1 Creating an ECMAScript in a Driver

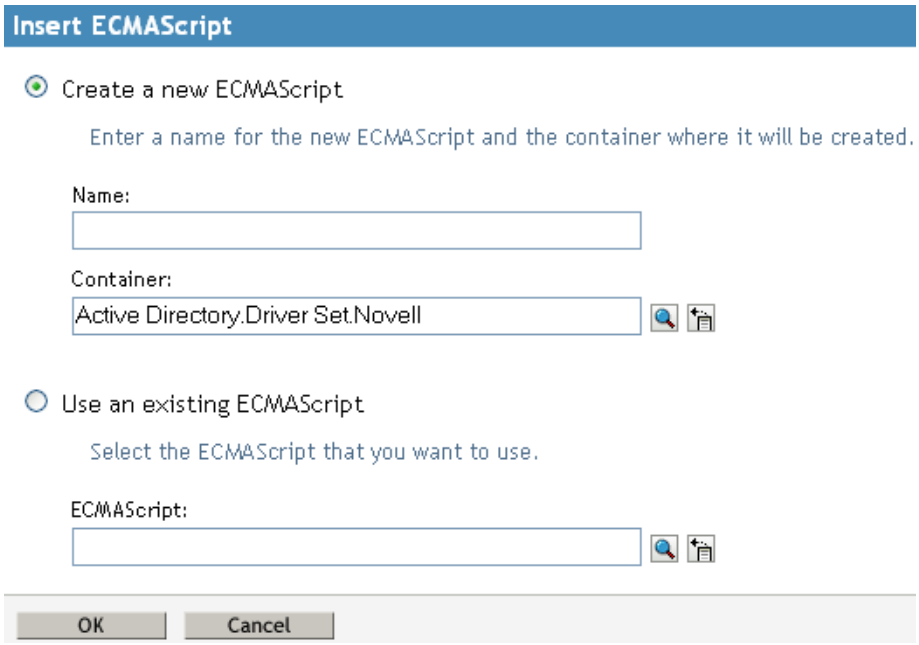
1 Access the Identity Manager Driver Overview by following the steps in "Accessing the Identity Manager Driver Overview Page" on page 270.

Ensure that the driver where you want to create the ECMAScript is the driver displayed in the Identity Manager Driver Overview.

2 Select *Advanced* > *ECMAScript*, then click *Insert*.



3 Select *Create a new ECMAScript*.



4 Specify the name of the ECMAScript.

5 Browse to and select the driver where you want to store the ECMAScript, then click *OK*.

6 Click *Enable ECMAScript editing*, then type the ECMAScript.

If you have an existing ECMAScript in a file, you want to use, open the file in a text editor and copy the information into the ECMAScript editor.

7 Click *Apply* to save the information in the ECMAScript editor

or

Click *OK* to save the changes and close the ECMAScript editor.

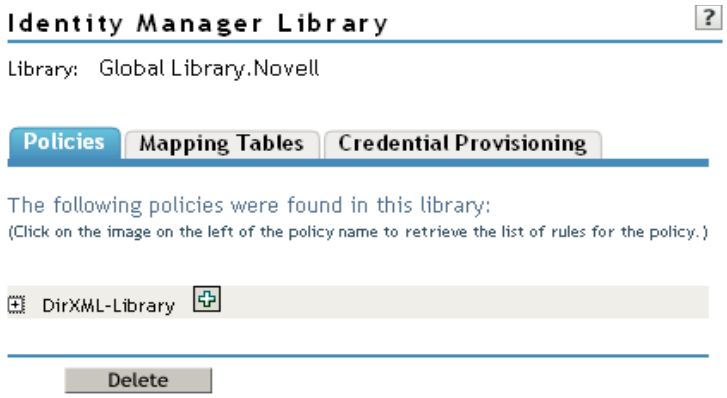
8.1.2 Creating an ECMAScript in a Library

1 Access the Identity Manager Driver Set Overview page by following the steps in "Accessing the Identity Manager Driver Set Overview Page" on page 269.

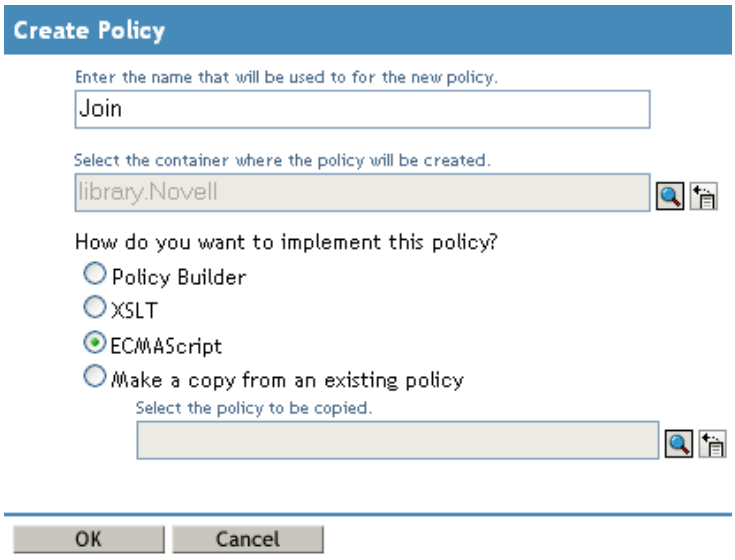
2 Click the *Libraries* tab.

3 Click the library you want to add an ECMAScript to.

4 Click the *Policies* tab, then click the plus icon.



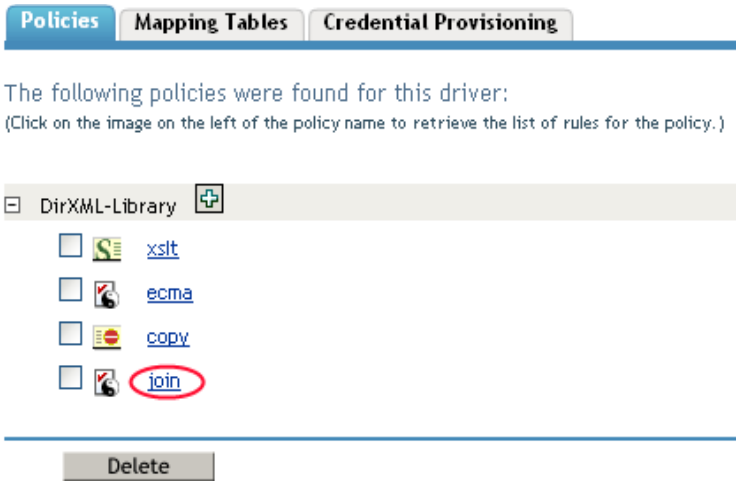
5 Click the *Create a policy in this container icon*.



6 Specify the name for the ECMAScript.

7 Select *ECMAScript*, then click *OK*.

8 Click the ECMAScript in the list of policies stored in the library.



9 On *Identity Manager*, click *Edit Resource* > select *Enable ECMAScript editing*, then type the ECMAScript.

If you have an existing ECMAScript in a file that you want to use, open the file in a text editor and copy the information into the ECMAScript editor.

10 Click *Apply* to save the information in the ECMAScript editor

or

Click *OK* to save the changes and close the ECMAScript editor.

8.2 Using an Existing ECMAScript

If you have an existing ECMAScript in Identity Manager, you can copy the object to a new location. The existing ECMAScript can be copied to a driver or a library.

- Section 8.2.1, "Using an Existing ECMAScript in a Driver," on page 86
- Section 8.2.2, "Using an Existing ECMAScript in a Library," on page 86

8.2.1 Using an Existing ECMAScript in a Driver

1 Access the Identity Manager Driver Overview page by following the steps in "Accessing the Identity Manager Driver Overview Page" on page 270.

Ensure that the driver where you want to copy the existing ECMAScript to is the driver displayed in the Identity Manager Driver Overview.

2 Select *Advanced* > *ECMAScript*, then click *Insert*.

3 Select *Use an existing ECMAScript*.

4 Browse to and select the existing ECMAScript.

5 Click *OK*.

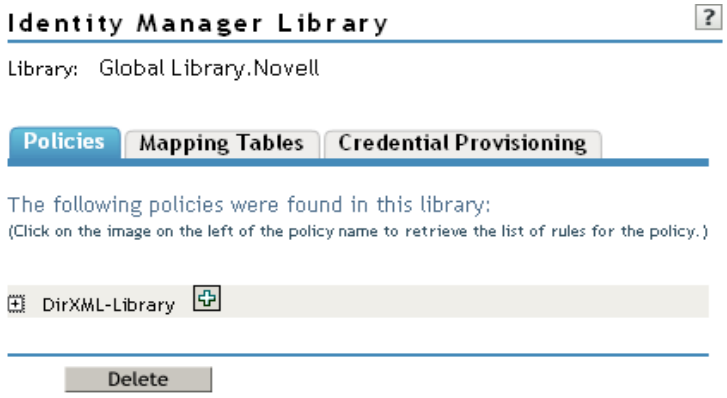
8.2.2 Using an Existing ECMAScript in a Library

1 Access the Identity Manager Driver Set Overview page by following the steps in "Accessing the Identity Manager Driver Set Overview Page" on page 269.

2 Click the *Libraries* tab.

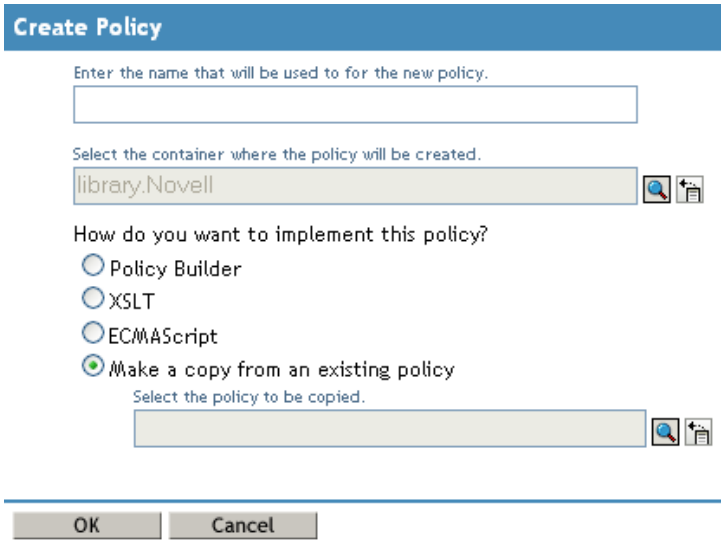
3 Click the library you want to add the existing ECMAScript to.

4 Click the *Policies* tab, then click the plus icon.



5 Select *Make a copy from an existing policy*.

6 Browse to and select the existing ECMAScript, then click *OK*.



8.3 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.

8.3.1 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 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>
```

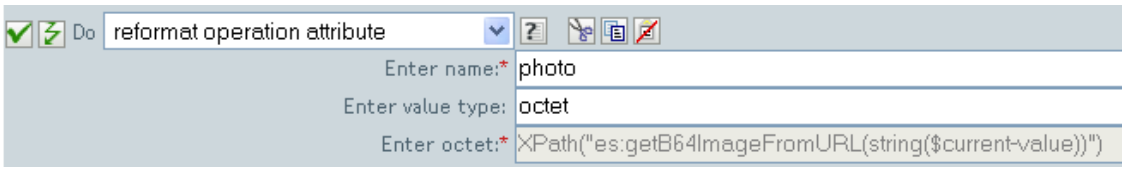
Function: `<static> String getB64ImageFromURL(<String> urlString)`

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 8-1 *Reformat Photo Example*



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

8.3.2 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 is used as an Input Transformation or Output Transformation policy.

NOTE: DirXML Script has the split and join functionality built in, but XSLT does not. This type of function allows XSLT to have the split and join functionality.

There are two functions:

- "Join" on page 89
- "Split" on page 89

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.

8.3.3 XSLT Policy Calling an ECMAScript Function in the Style Sheet

The XSLT policy demonstrates embedding an ECMAScript function definition with the XSLT style sheet. The function converts 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.

Conditions

9

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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Comparison Mode

Some condition tests have a mode parameter that indicates how the comparison is done.

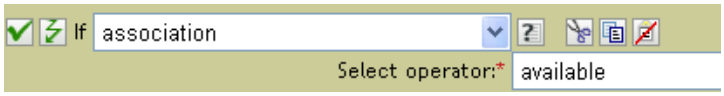
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. 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 have a comparison mode parameter are:

- Equal
- Not Equal
- 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.



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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Comparison Mode

Some condition tests have a mode parameter that indicates how the comparison is done.

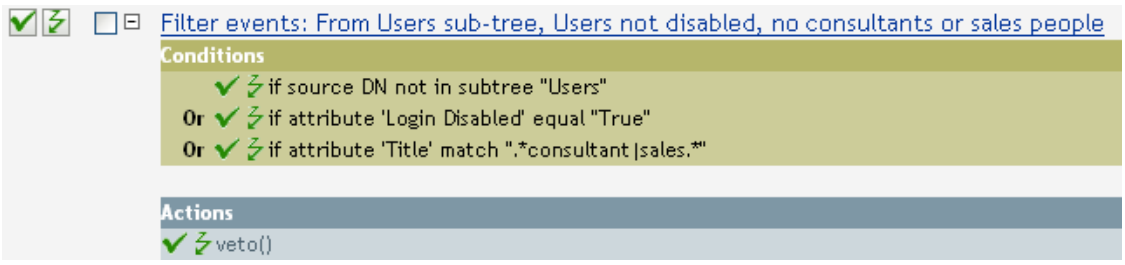
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. 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:

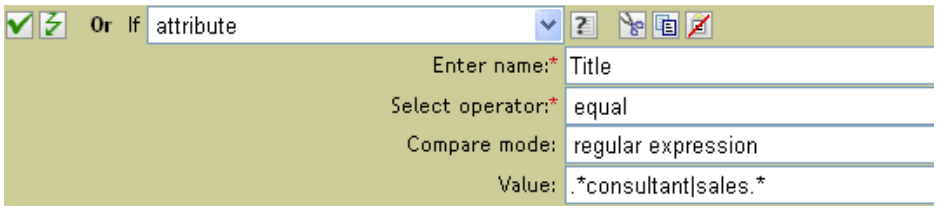
- Equal
- Not Equal
- Not Greater Than
- Less Than
- Not Less Than

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 001-Event-FilterByContainerDisabledOrTitle.xml (../samples/001-Event-FilterByContainerDisabledOrTitle.xml).



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


Or If attribute
Enter name:*
Title
Select operator:*
equal
Compare 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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Comparison Mode

Some condition tests have a mode parameter that indicates how the comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.

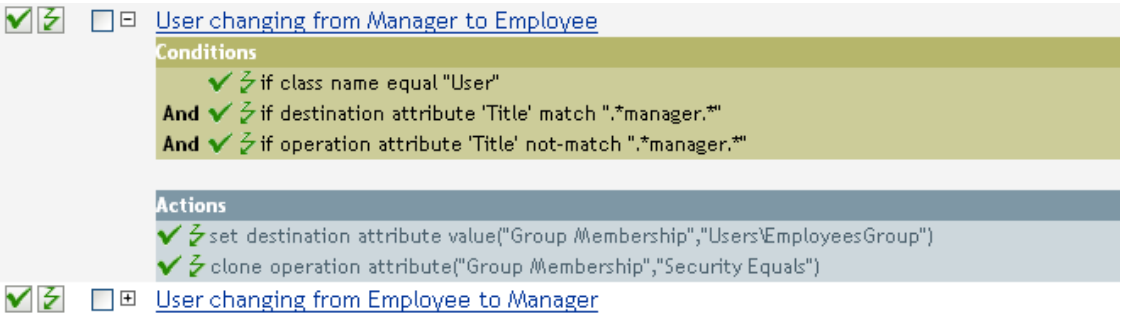
Mode	Description
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, 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:

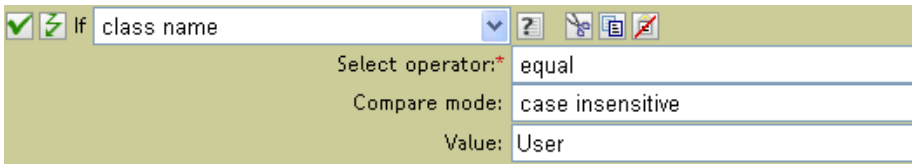
- Equal
- Not Equal
- Not Greater Than
- Less Than
- Not Less Than

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 004-Command-GroupChangeOnTitleChange.xml (../samples/004-Command-GroupChangeOnTitleChange.xml).



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


If
class name
Select operator:*
equal
Compare 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 Greater 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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Comparison Mode

Some condition tests have a mode parameter that indicates how the comparison is done.

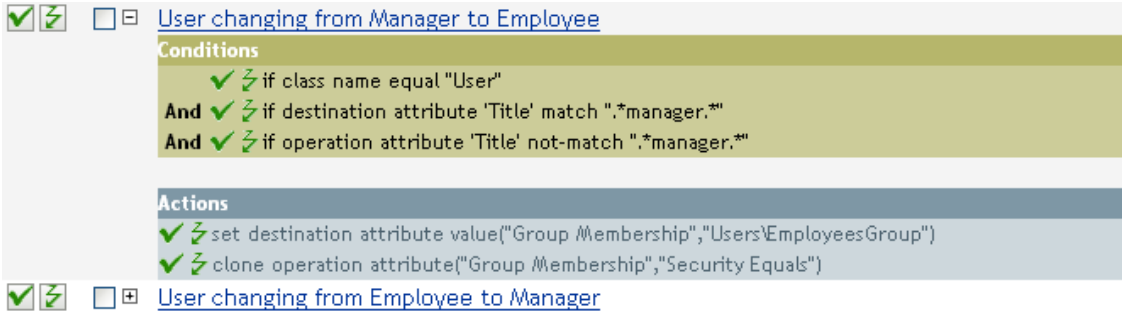
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. 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:

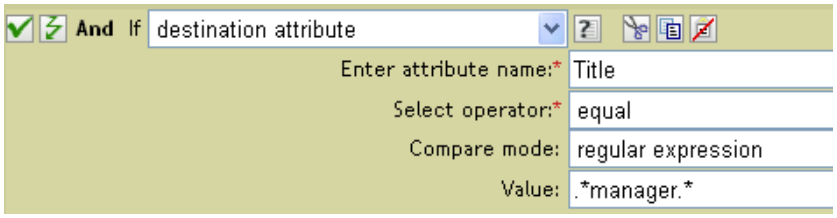
- Equal
- Not Equal
- Not Greater Than
- Less Than
- Not Less Than

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 004-CommandGroupChangeOnTitleChange.xml (../samples/004-Command-GroupChangeOnTitleChange.xml).



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

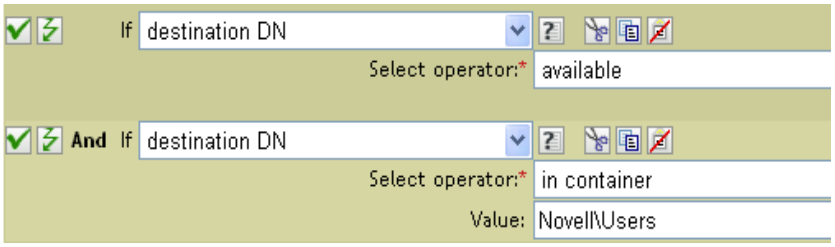
Operator	Returns True when...
Available	There is a destination DN available.
Not Available	Available would return False.
Equal	There is a destination DN available, and it equals the specified value when compared using semantics appropriate to the DN format of the destination data store.
Not Equal	Equal 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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Example


If
destination DN
Select operator:*
available
And
If
destination DN
Select operator:*
in container
Value:
Novell\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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). 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

Some condition tests have a mode parameter that indicates how the comparison is done.

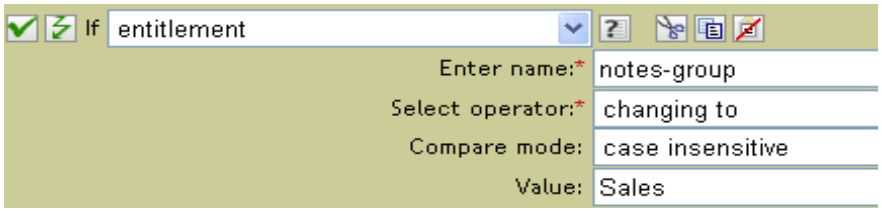
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. 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



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/b6yi6f6.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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

- Not Less Than

Comparison Mode

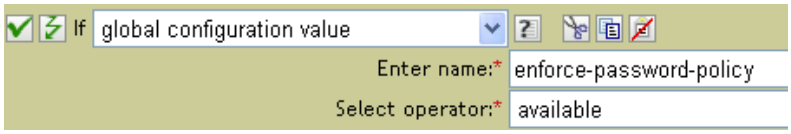
Some condition tests have a mode parameter that indicates how the comparison is done.

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



If Local Variable

Performs a test on a local variable. The test performed depends on the specified operator.

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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Comparison Mode

Some condition tests have a mode parameter that indicates how the comparison is done.

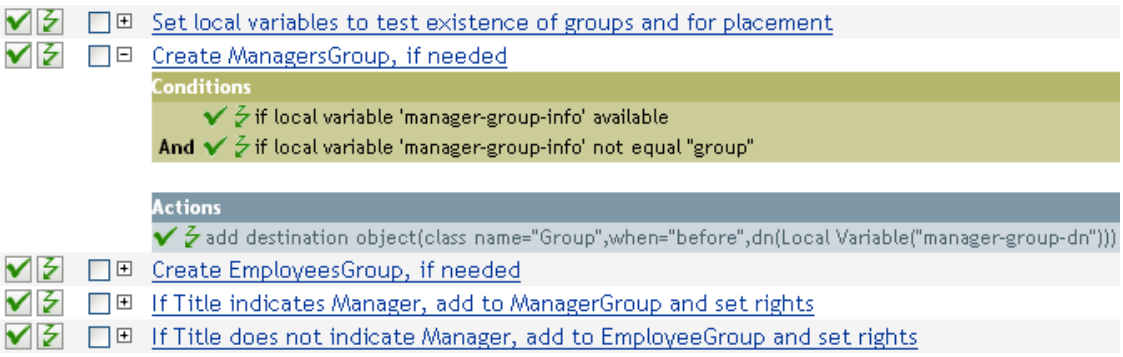
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. 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:

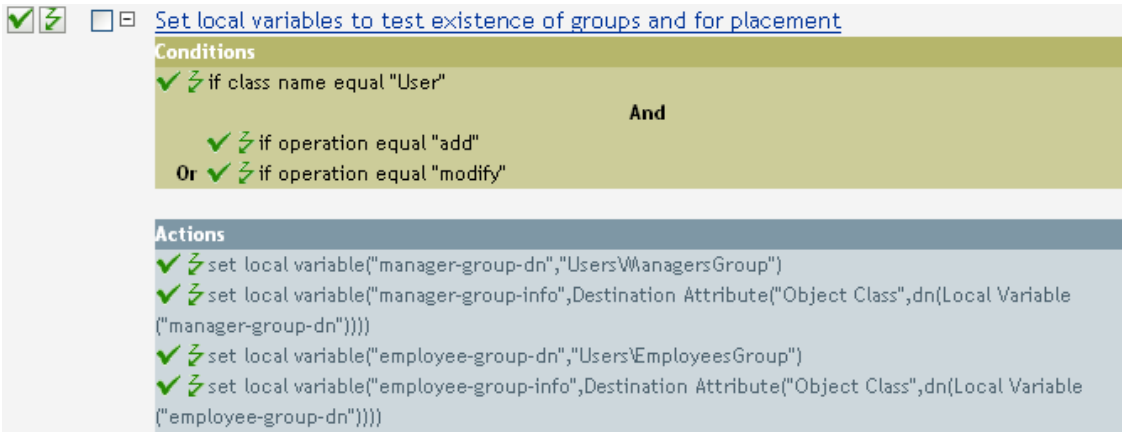
- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Example

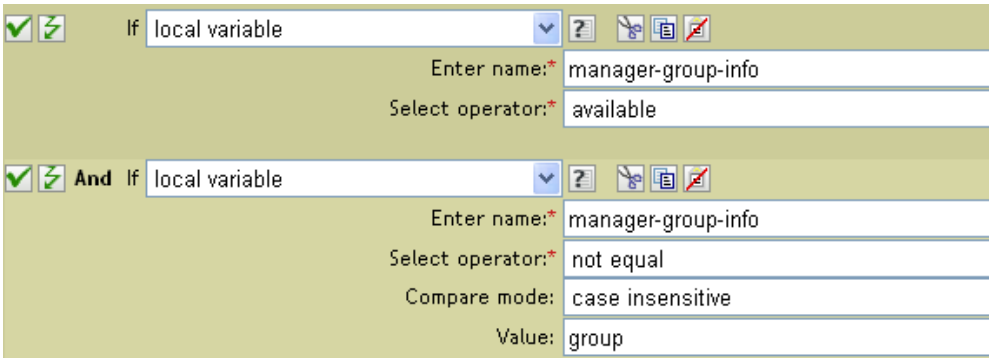
The example adds a User object to the appropriate group, Employee or Manager, based on Title. It also creates the group, if needed, and sets up security equal to that group. The policy is Govern Groups for User Based on Title Attribute, and it is available for download from the Novell Support Web site. For more information, see "Downloading Identity Manager Policies" in *Understanding Policies for Identity Manager 3.6*.To view the policy in XML, see 003-Command-AddCreate-Groups.xml (../samples/003-Command-AddCreateGroups.xml).



The policy contains five rules that are dependent on each other.



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.



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

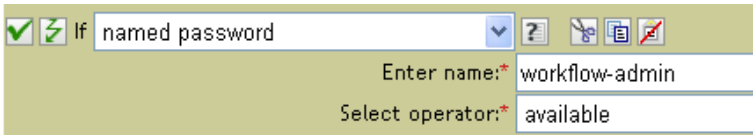
Specify the name of the named password to test for the selected condition.

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



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...
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 or 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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). 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

Some condition tests have a mode parameter that indicates how the 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.

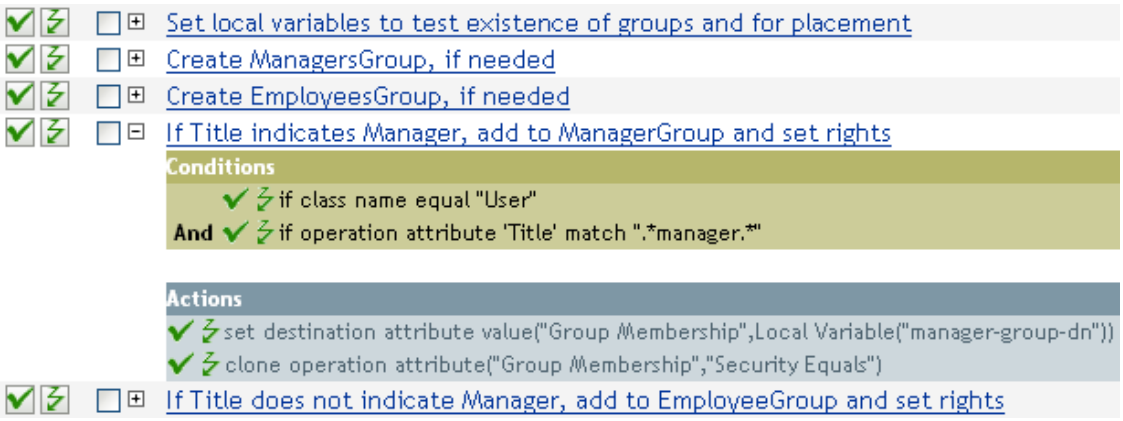
The operators that contain the comparison mode parameter are:

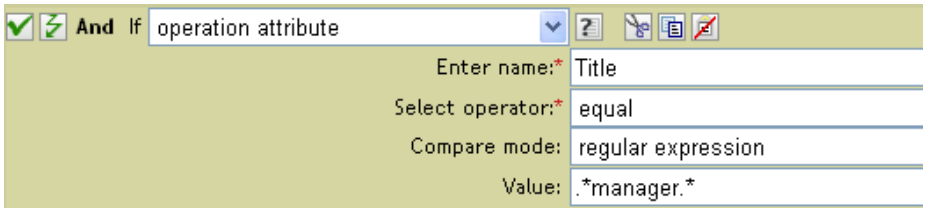
- Equal
- Not Equal
- Changing To
- Changing From

- 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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 003-Command-Add-CreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).





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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Comparison Mode

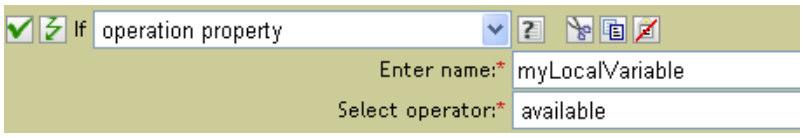
Some condition tests have a mode parameter that indicates how the comparison is done.

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



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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). 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
- delete
- generated-password
- get-named-password
- modify
- modify-association
- modify-password

- move
- init-params
- instance
- 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.

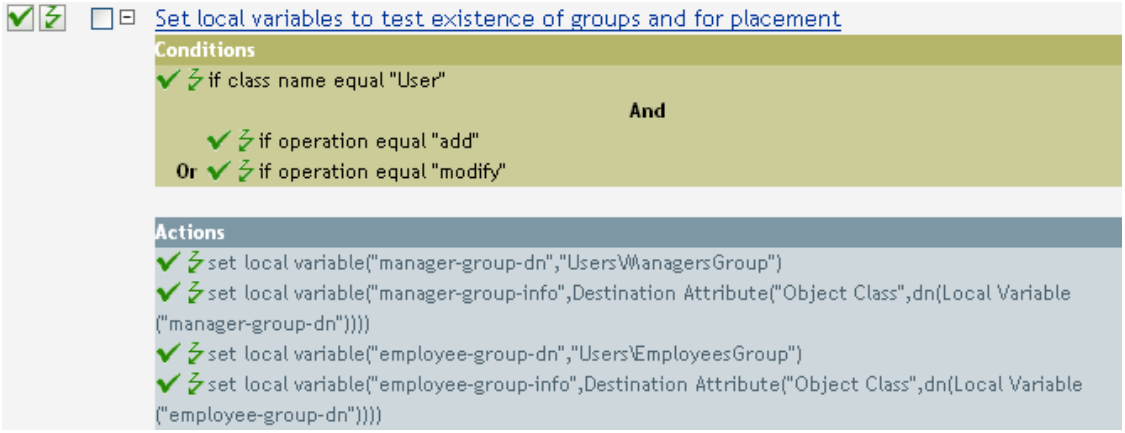
Comparison Mode

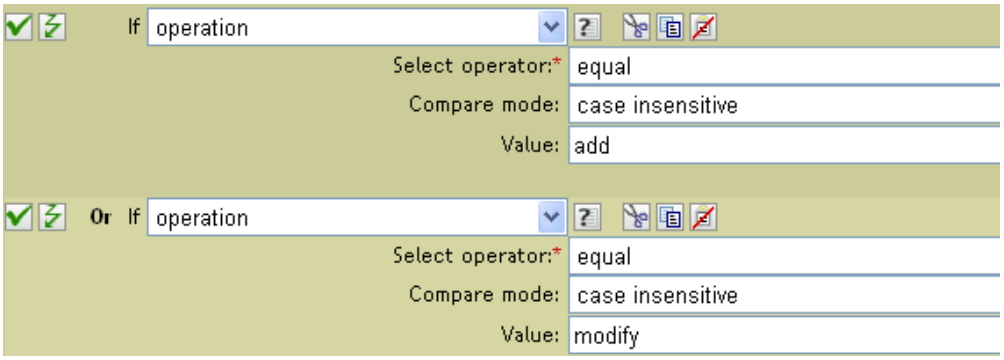
Some condition tests have a mode parameter that indicates how the comparison is done.

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.

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 003-Command-AddCreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).





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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Comparison Mode

Some condition tests have a mode parameter that indicates how the 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.

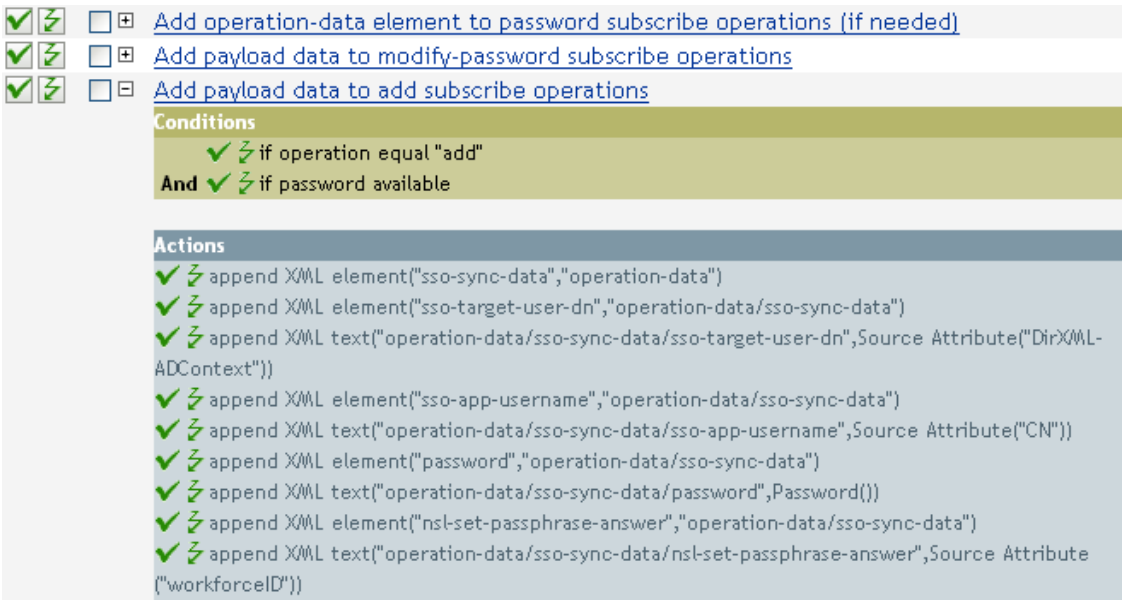
The operators that contain the comparison mode parameter are:

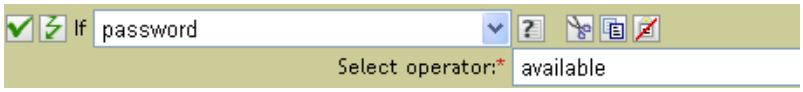
- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Example

If you are implementing *Novell Credential Provisioning Policies for Identity Manager 3.6*, 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" in *Novell Credential Provisioning Policies for Identity Manager 3.6*. 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
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"))


If
password
Select 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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Comparison Mode

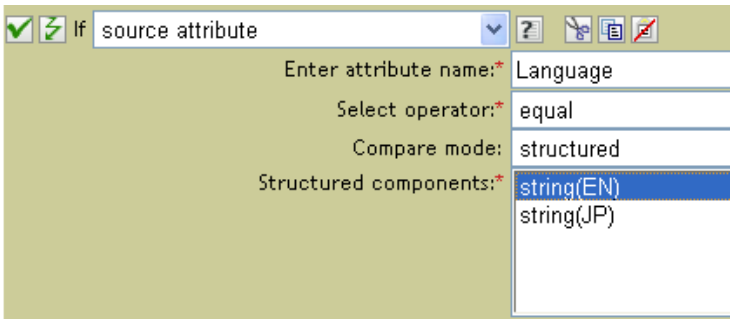
Some condition tests have a mode parameter that indicates how the comparison is done.

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



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.

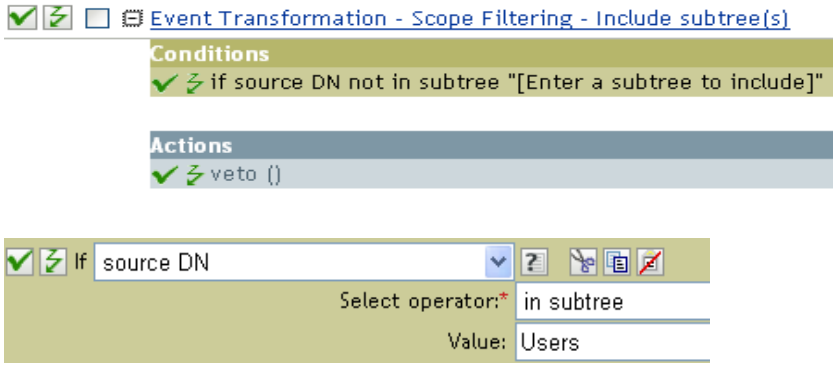
Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). 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” on page 56. To view the policy in XML, see predef_transformation_filter_exclude_subtrees.xml (../samples/predef_transformation_filter_exclude_subtrees.xml).



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 selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 133). The operators that contain the value field are:

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

Comparison Mode

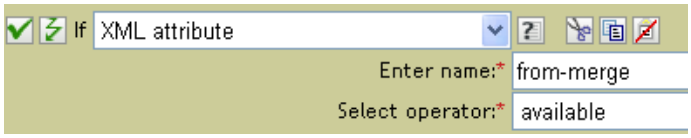
Some condition tests have a mode parameter that indicates how the comparison is done.

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



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.

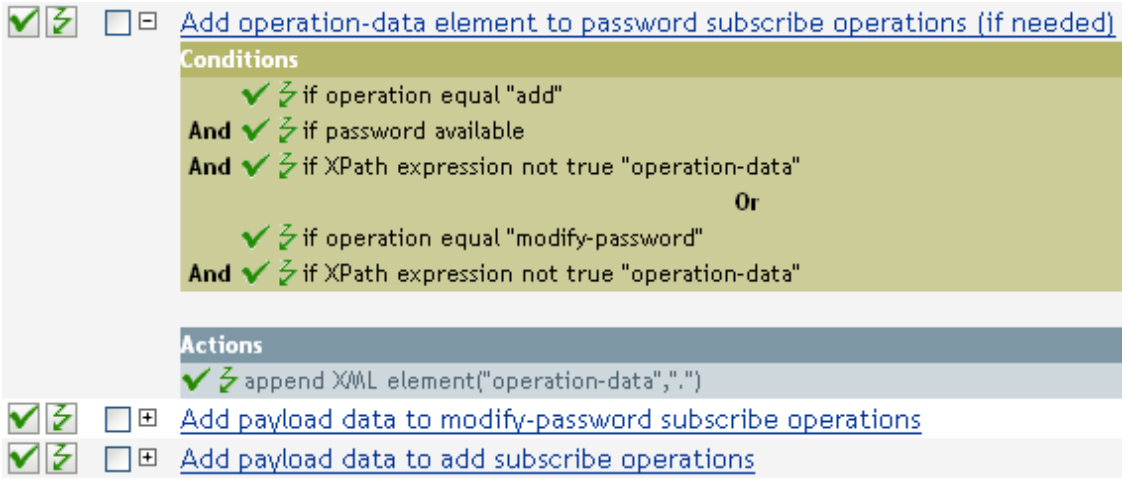
Remarks

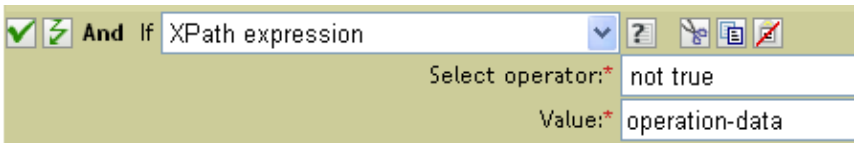
For more information on using XPath expression with policies, see "XPath 1.0 Expressions" in the *Understanding Policies for Identity Manager 3.6*.

Example

If you are implementing *Novell Credential Provisioning Policies for Identity Manager 3.6*, 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" in *Novell Credential Provisioning Policies for Identity Manager 3.6*. 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.




And
If
XPath expression
Select 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 from $<variable-name>$ is replaced with the value of the local or global variable with the five name. $<variable-name>$ must be a legal variable name. For more information on what is a legal XML name, see W3C Extensible Markup Language (http://www.w3.org/TR/2006/REC-xml11-20060816/#sec-suggested-names).

Actions

10

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.

Add Association

Sends an add association command with the specified association to the Identity Vault.

Fields

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

DN

Specify the DN of the target object or leave the field blank to use the current object.

Association

Specify the value of the association to be added.

Example

Add Destination Attribute Value

Adds a value to an attribute on an object in the destination data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 207).

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 207).

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

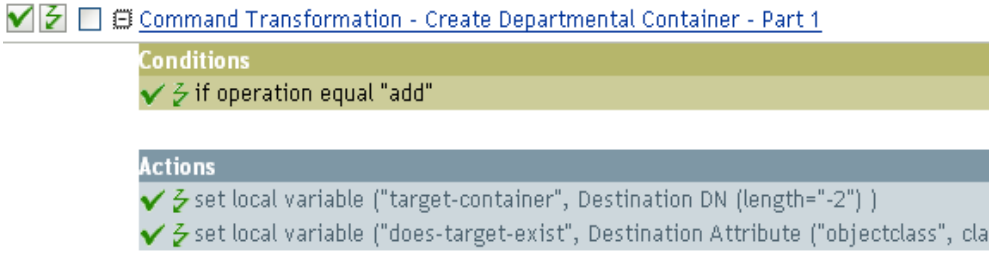
Select the syntax of the attribute value to be added. The options are string, counter, dn, int, interval, octet, state, structured, teleNumber, or time.

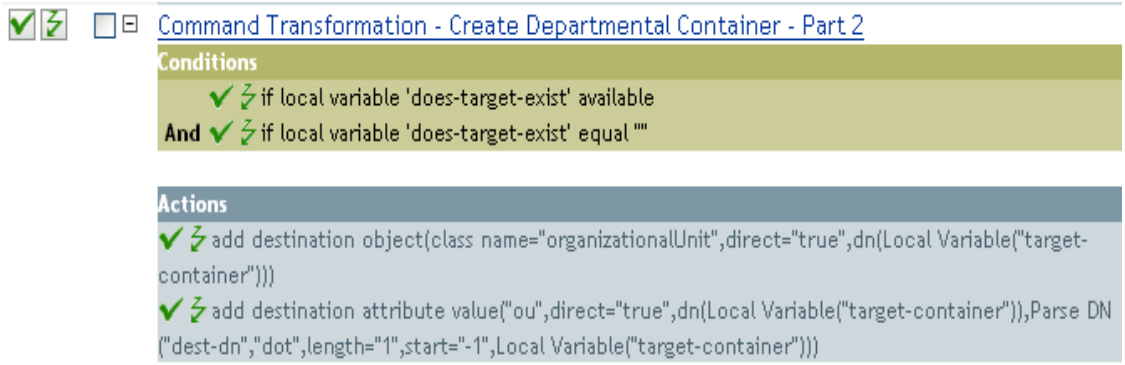
Value

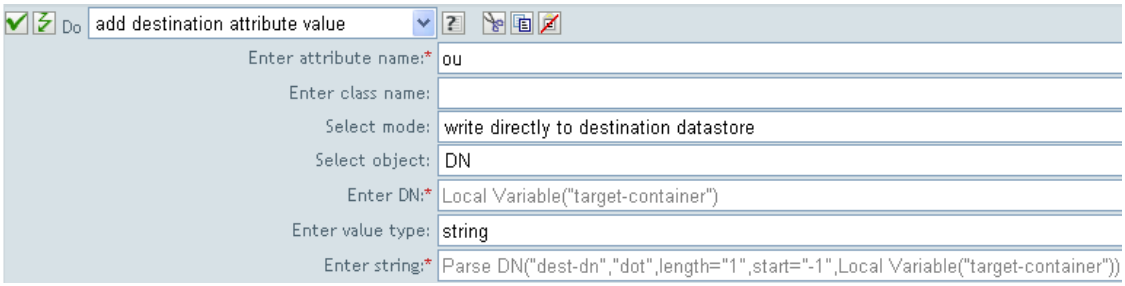
Specify the attribute value to be added.

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 Part 2" on page 48. 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 2
Conditions
if local variable 'does-target-exist' available
And if local variable 'does-target-exist' equal ""
Actions
add destination object(class name="organizationalUnit",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
Enter attribute name:*
ou
Enter class name:
Select mode:
write directly to destination datastore
Select object:
DN
Enter DN:*
Local Variable("target-container")
Enter value type:
string
Enter string:*
Parse DN("dest-dn","dot",length="1",start="-1",Local Variable("target-container"))

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 207).

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

DN

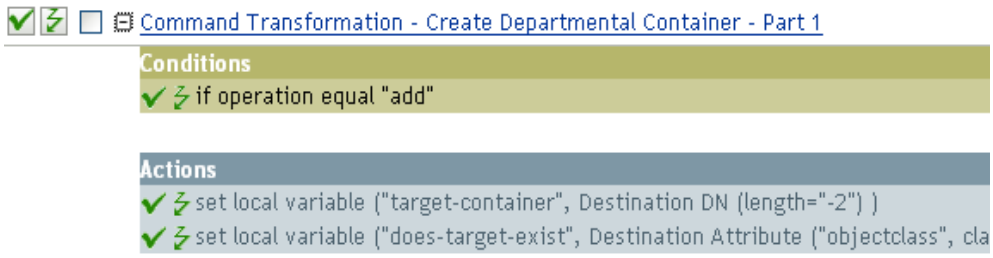
Specify the DN of the object to be created.

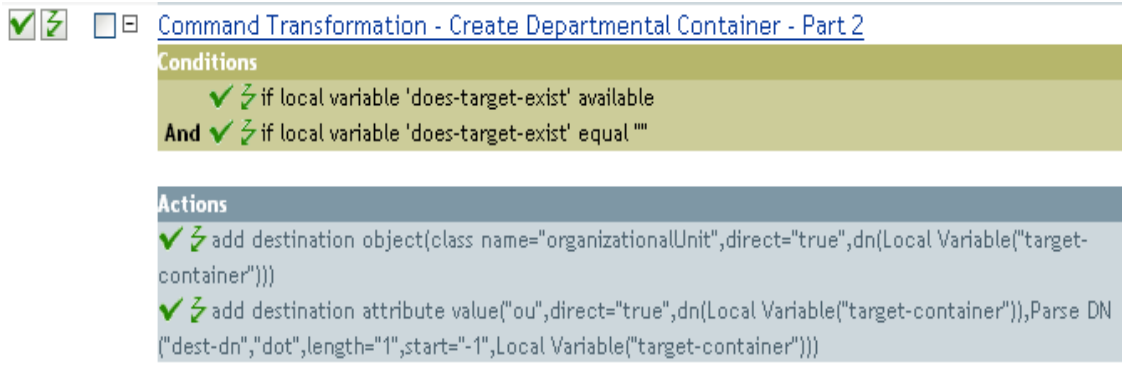
Remarks

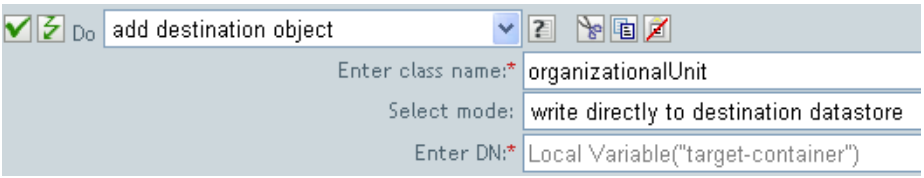
Any attribute values to be added as part of the object creation must be done in subsequent Add Destination Attribute Value actions using the same DN.

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 Part 2" on page 48 from the predefined rules. To see the policy in XML, see predef_command_create_dept_container1.xml (../samples/predef_command_create_dept_container1.xml) and predef_command_create_dept_container2.xml (../samples/predef_command_create_dept_container2.xml).







The OU object is created. The value for the OU attribute is created from the destination attribute value action that occurs after this action.

Add 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 207).

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 207).

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

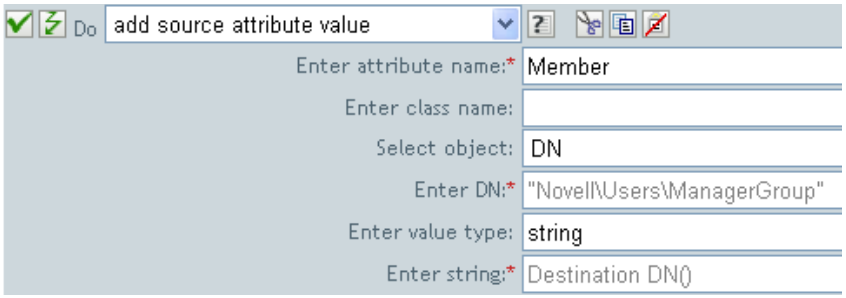
Value Type

Select the syntax of the attribute value to be added. The options are string, counter, dn, int, interval, octet, state, structured, teleNumber, or time.

String

Specify the attribute value to be added.

Example



Add Source Object

Creates an object of the specified type in the source data store, with the name and location provided in the DN field. Any attribute values to be added as part of the object creation must be done in subsequent Add Source Attribute Value (page 142) actions using the same DN.

Fields

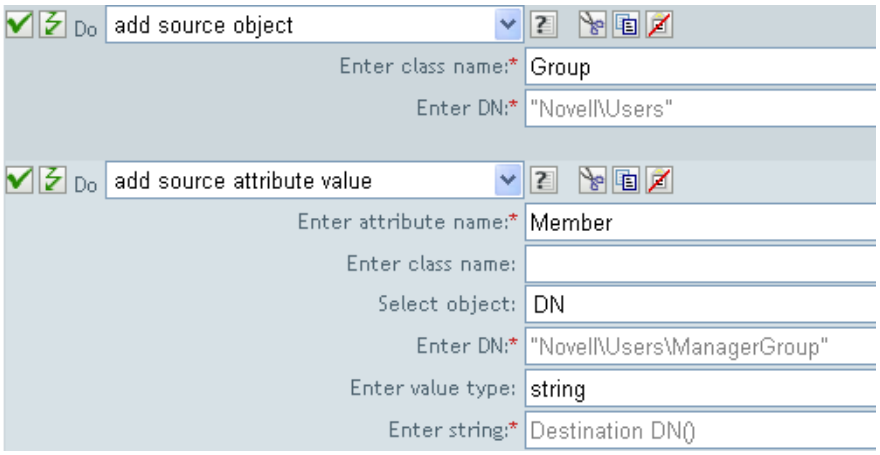
Class Name

Specify the class name of the object to be added. Supports variable expansion. For more information, see Variable Expansion (page 207).

DN

Specify the DN of the object to be added.

Example



Append XML Element

Appends a custom element, with the name specified in the *Name* field, to the set of elements selected by the XPath expression. If *Before XPath Expression* is not specified, the new element is appended after any existing children of the selected elements. If *Before XPath Expression* is specified, it is evaluated relative to each of the elements selected by the expression to determine which of the children to insert before. If *Before XPath Expression* evaluates to an empty node set or a node set that does not contain any children of the selected element, the new element is appended after any existing children; otherwise, the new element is inserted before each of the nodes in the node set selected by before that are children of the selected node.

Fields

Name

Specify the tag name of the XML element. This name can contain a namespace prefix if the prefix has been previously defined in this policy. Supports variable expansion. For more information, see Variable Expansion (page 207).

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.

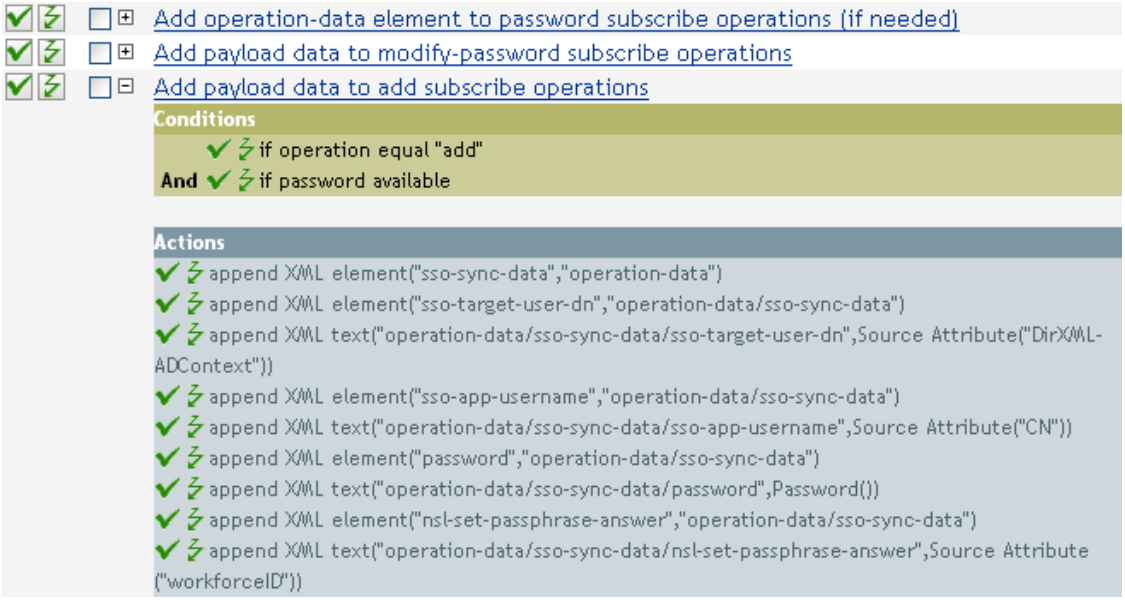
Remarks

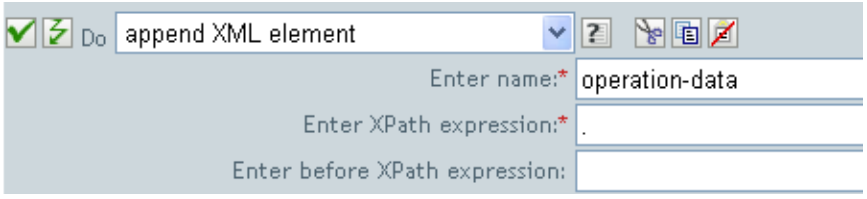
For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *Understanding Policies for Identity Manager 3.6*.

Example

If you are implementing *Novell Credential Provisioning Policies for Identity Manager 3.6*, 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" in *Novell Credential Provisioning Policies for Identity Manager 3.6*. 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
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 name:*
operation-data
Enter XPath expression:*
.
Enter before 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.

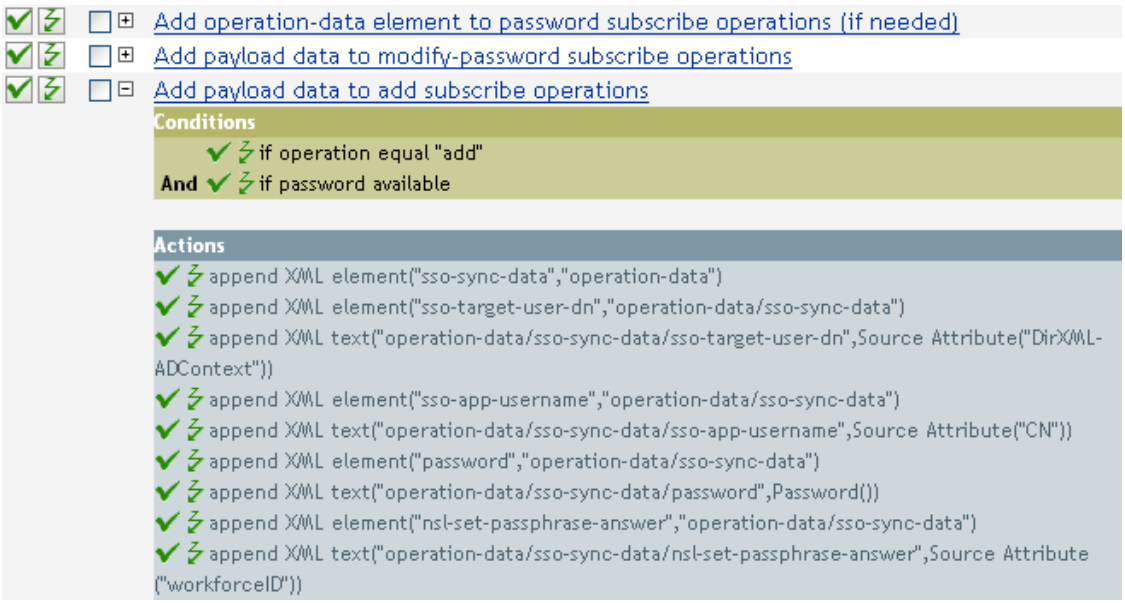
Remarks

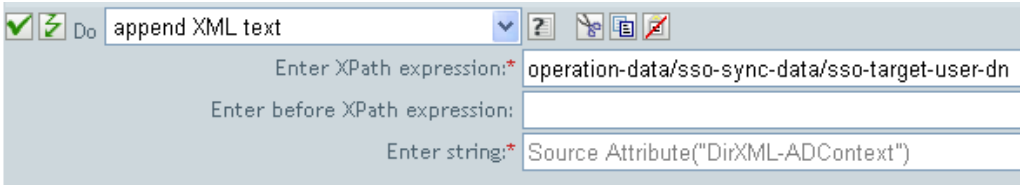
For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *Understanding Policies for Identity Manager 3.6*.

Example

If you are implementing *Novell Credential Provisioning Policies for Identity Manager 3.6*, 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" in *Novell Credential Provisioning Policies for Identity Manager 3.6*. 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
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
Enter XPath expression:*
operation-data/sso-sync-data/sso-target-user-dn
Enter before XPath expression:
Enter string:*
Source Attribute("DirXML-ADContext")

Break

Ends processing of the current operation by the current policy.

Example

Clear Destination Attribute Value

Removes all values for the named attribute from an object in the destination data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 207).

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 207).

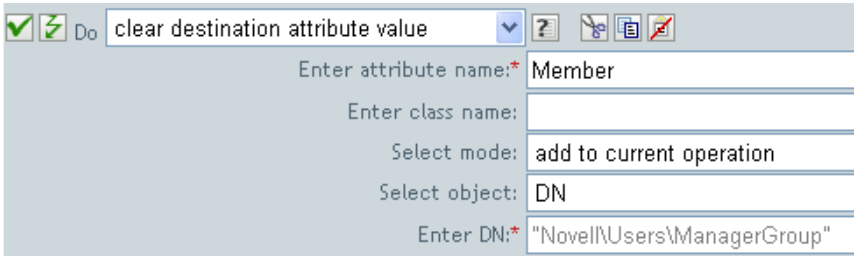
Mode

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.

Example



Clear Operation Property

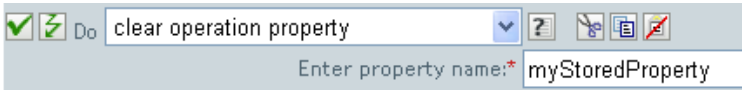
Clears any operation property with the provided name from the current operation. The operation property is the XML attribute attached to an `<operation-data>` element by a policy. An XML attribute is a name/value pair associated with an element in the XDS document.

Fields

Property Name

Specify the name of the operation property to clear. Supports variable expansion. For more information, see Variable Expansion (page 207).

Example



Clear SSO Credential

Clears the Single Sign On credential so objects can be deprovisioned. Additional information about the credential to be cleared can be 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 for Identity Manager 3.6*.

Fields

Credential Store Object DN

Specify the DN of the repository object. Supports variable expansion. For more information, see Variable Expansion (page 207).

Target User DN

Specify the DN of the target users.

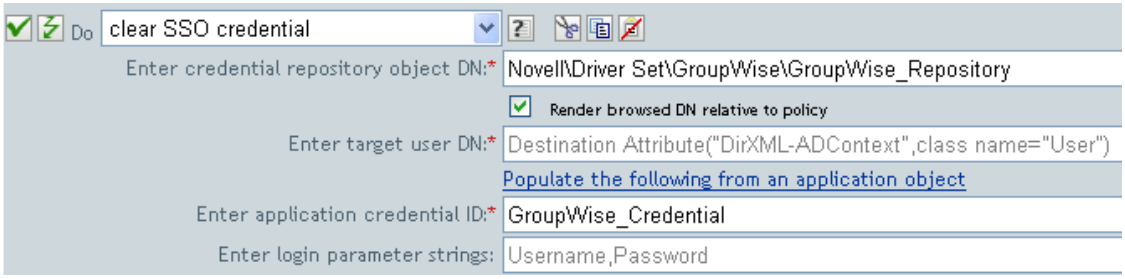
Application Credential ID

Specify the application credential that is stored in the application object. Supports variable expansion. For more information, see Variable Expansion (page 207).

Login Parameter Strings

Specify each login parameter for the application. The login parameters are the authentication keys stored in the application object.

Example



Clear Source Attribute Value

Removes all values of an attribute from an object in the source data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. Supports variable expansion. For more information, see Variable Expansion (page 207).

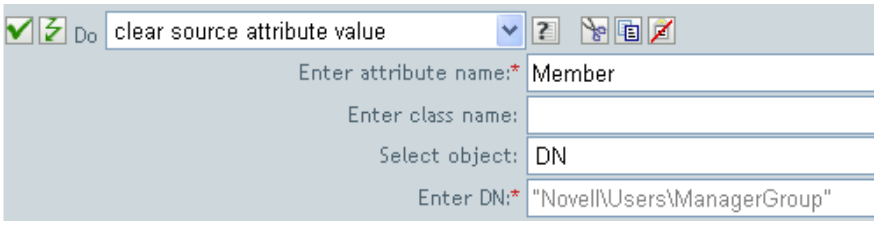
Class Name

(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 207).

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

Example



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.

Before XPath Expression

Specify the XPath 1.0 expression that evaluates relative to each of the nodes selected by the destination XPath expression that returns a node set containing the child nodes that the non-attribute cloned nodes should be inserted before.

Remarks

For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *Understanding Policies for Identity Manager 3.6*.

Example

Clone Operation Attribute

Copies all occurrences of an attribute within the current operation to a different attribute within the current operation.

Fields

Source Name

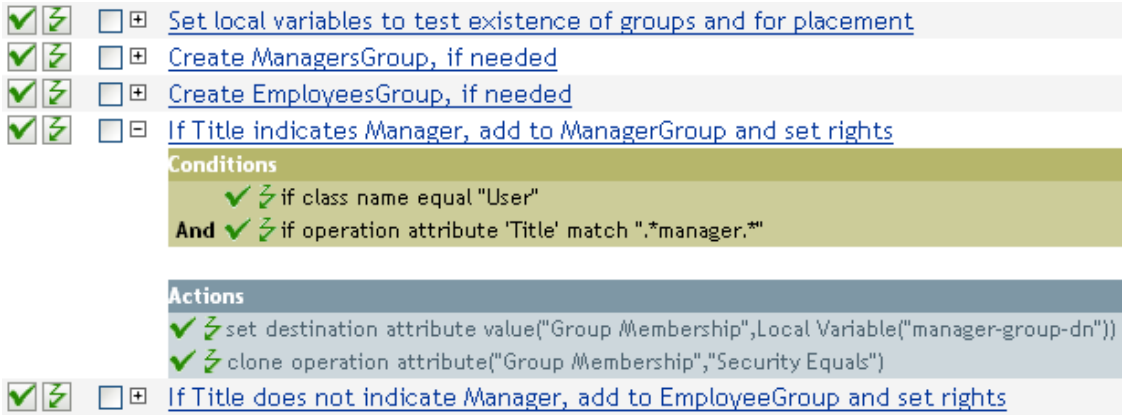
Specify the name of the attribute to be copied from. Supports variable expansion. For more information, see Variable Expansion (page 207).

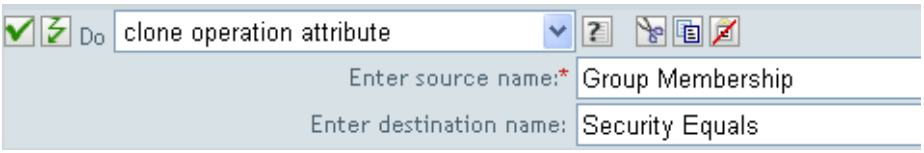
Destination Name

Specify the name of the attribute to be copied to. Supports variable expansion. For more information, see Variable Expansion (page 207).

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" in *Understanding Policies for Identity Manager 3.6*. To see the policy in XML, see 003-Command-AddCreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).





The Clone Operation Attribute is taking the information from the Group Membership attribute and adding 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.

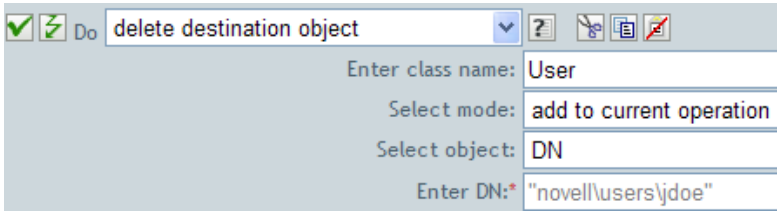
Mode

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 to delete in the destination data store. This object can be the current object, or it can be specified by a DN or an association.

Example



Delete Source Object

Deletes an object in the source data store.

Fields

Class Name

(Optional) Specify the class name of the object to delete in the source data store.

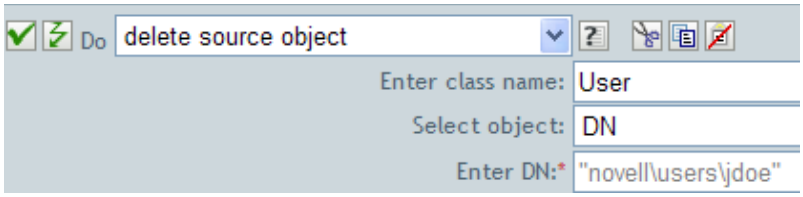
Object

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 the current object as the target object.

Example



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.

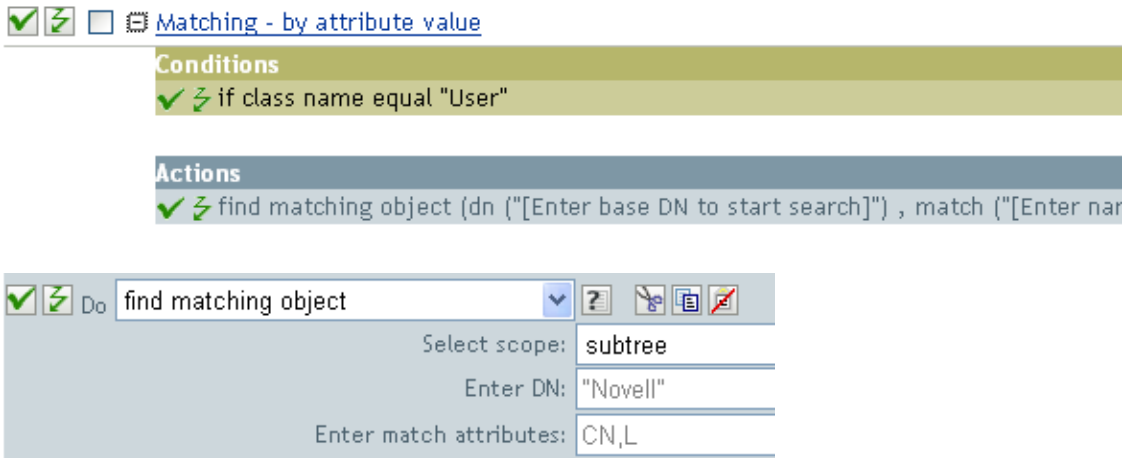
The DN argument is required when scope is “entry,” and is optional otherwise. At least one match attribute is required when scope is “subtree” or “subordinates.”

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 ￼. 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 Section 6.13, “Matching - By Attribute Value,” on page 62. To see the policy in XML, see predef_match_by_attribute.xml (../samples/predef_match_by_attribute.xml).



When you click the Argument Builder icon, the Match Attribute Builder comes up. You specify the attribute you want to match on in the builder. This example uses the CN and L attributes.

The left fields store the attributes to match. The right fields allow you to specify to use the value from the current object to match or to use another value. If you select *Other Value*, there are multiple value types to specify:

- counter
- dn
- int
- interval
- octet
- state
- string
- structured
- teleNumber
- time

For Each

Repeats a set of actions for each node in a node set.

Fields

Node Set

Specify the node set.

Action

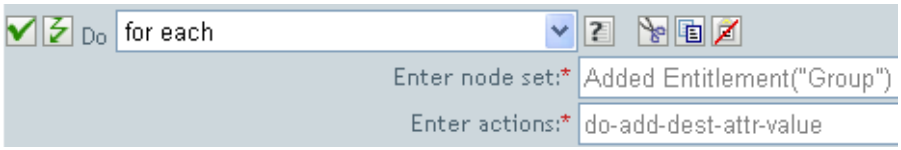
Specify the actions to perform on each node in the node set.

Remarks

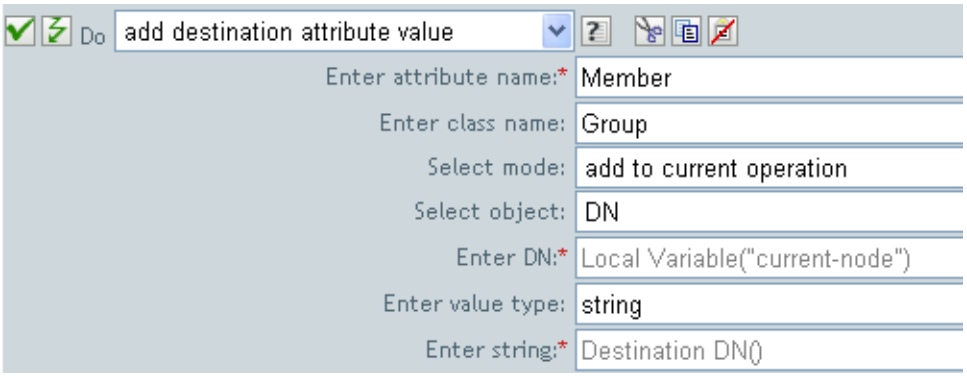
The current node is a different value for each iteration of the actions, if a local variable is used.

If the current node in the node set is an entitlement element, then the actions are marked as if they are also enclosed in an Implement Entitlement action. If the current node is a query element returned by a query, then that token is used to automatically retrieve and process the next batch of query results.

Example



The following is an example of the Argument Actions Builder, used to provide the action argument:



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 207).

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.

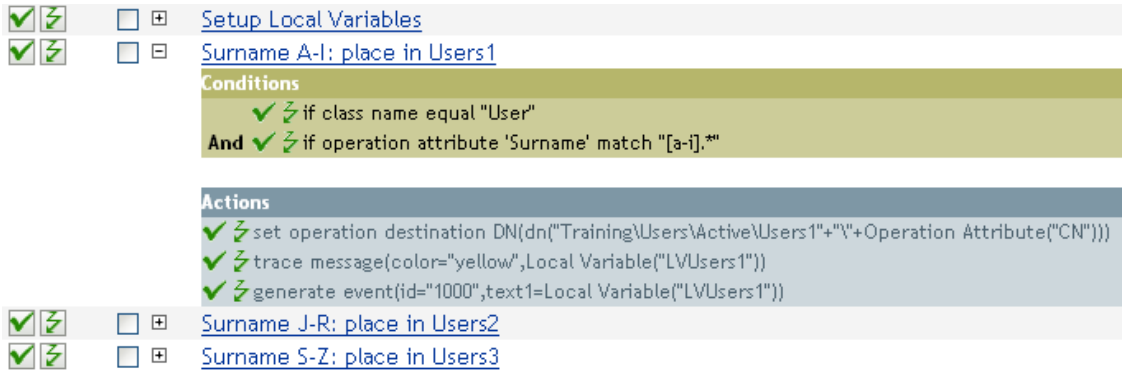
Tag	Description
text2	Text entered here is stored in the text2 event field.
text3	Text entered here is stored in the text3 event field.
value	Any number entered here is stored in the value event field.
value3	Any number entered here is stored in the value3 event field.
data	Data entered here is stored in the blob event field.

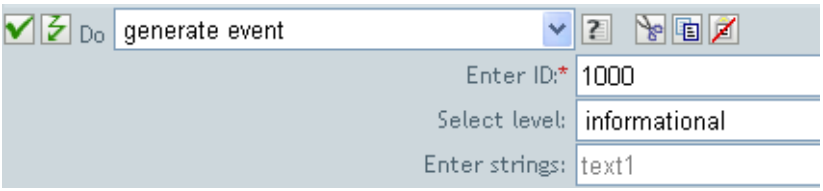
Remarks

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 001-Placement-BySurname.xml (../samples/001-Placement-BySurname.xml).





The following is an example of the Named String Builder, used to provide the strings argument.

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

Conditions

Specify the desired condition.

If Actions

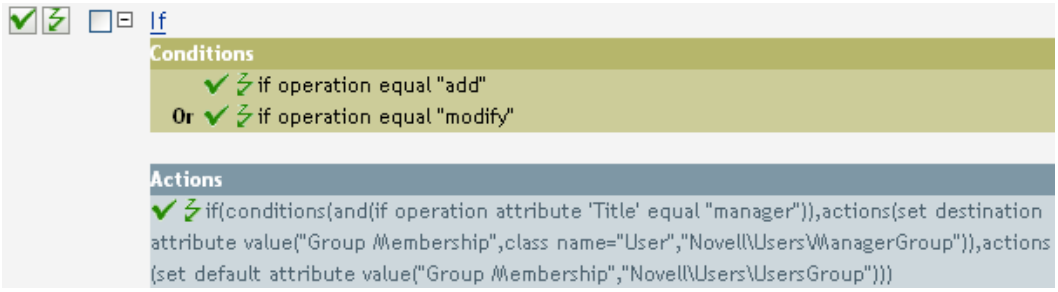
Specify the desired actions, if the conditions are True.

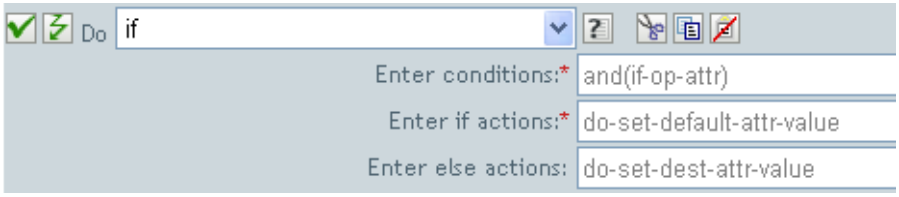
Else Actions

(Optional) Specify the desired actions, if the conditions are False.

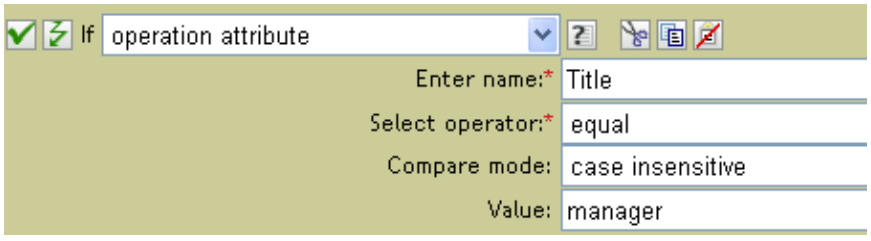
Example

During an add or modify operation, if the attribute of Title equals manager, the user object is added to the ManagerGroup group. If the Title does not equal manager, then the user object is added to the UsersGroup group. To view the policy in XML, see if.xml (../samples/if.xml).

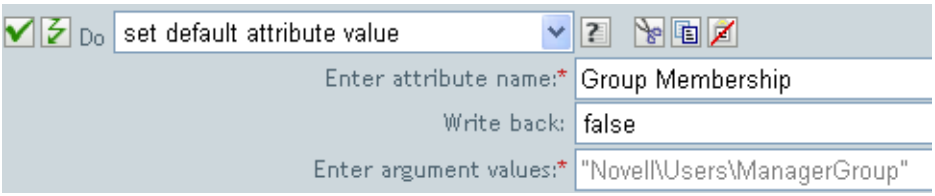




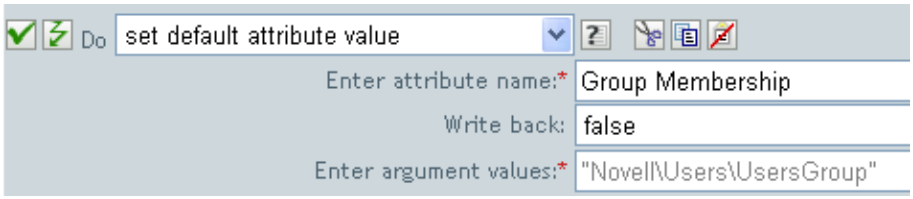
When you create the if action, you have to add a condition and one action. In this example there are two separate actions. The condition is if a user object has the title of manager.



The action is to add the user object to the ManagerGroup group.



If the title does not equal manager, the user object is placed in the UsersGroup group.



Implement Entitlement

Designates actions that implement an entitlement so that the status of those entitlements can be reported to the agent that granted or revoked the entitlement.

Fields

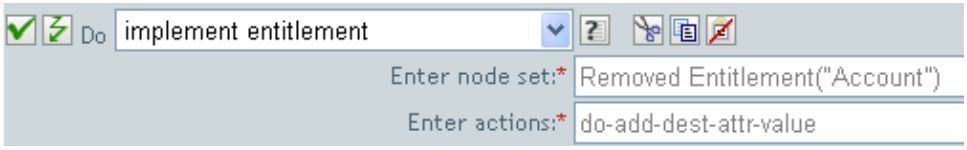
Node Set

Node set containing the entitlement being implemented by the specified actions.

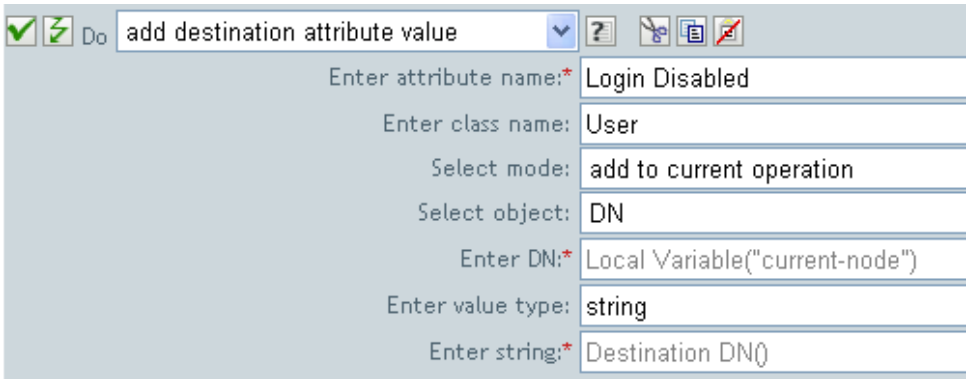
Action

Actions that implement the specified entitlements.

Example



The following is an example of the Argument Actions Builder, used to provide the action argument:



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

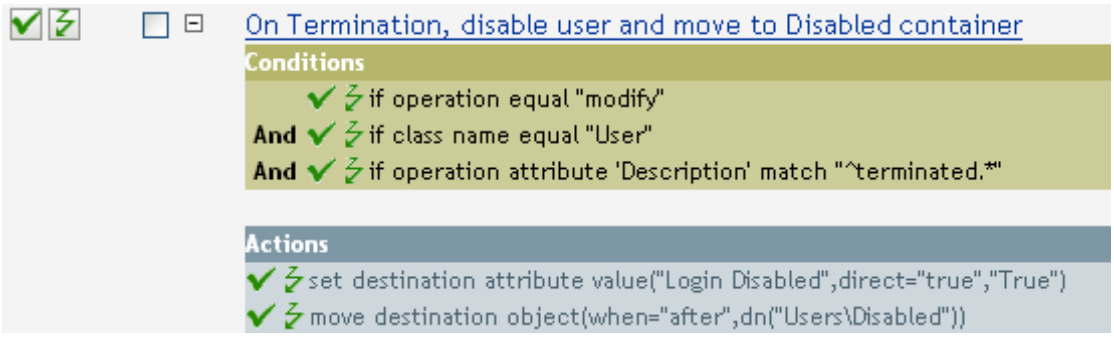
Select the container to receive the object. This container is specified by a DN or an association.

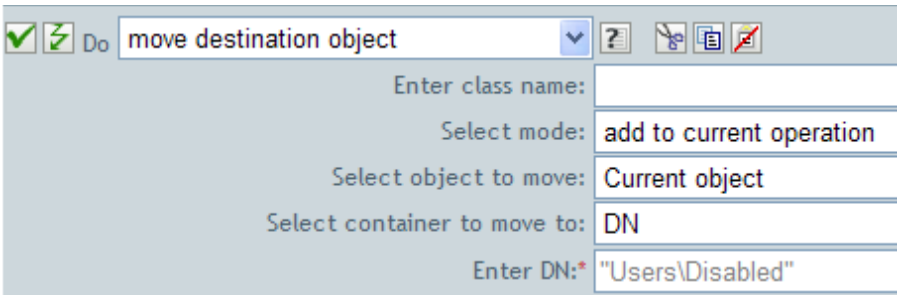
DN or Association

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" in *Understanding Policies for Identity Manager 3.6*. To view this policy in XML, see 005-Command-DisableMoveOnTermination (../samples/005-Command-DisableMoveOnTermination.xml).





The policy checks to see if it is a modify event on a User object and if the attribute Description contains the value of terminated. If that is the case, then it sets the attribute of Login Disabled to true and moves the object into the User\Disabled container.

Move Source Object

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.

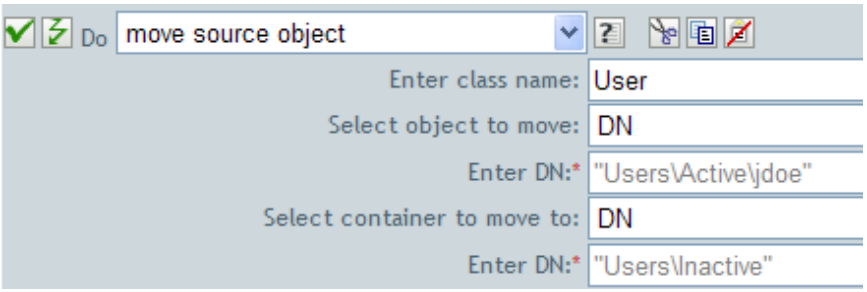
Object to Move

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



Reformat Operation Attribute

Reformats all values of an attribute within the current operation by using a pattern.

Fields

Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 207).

Value Type

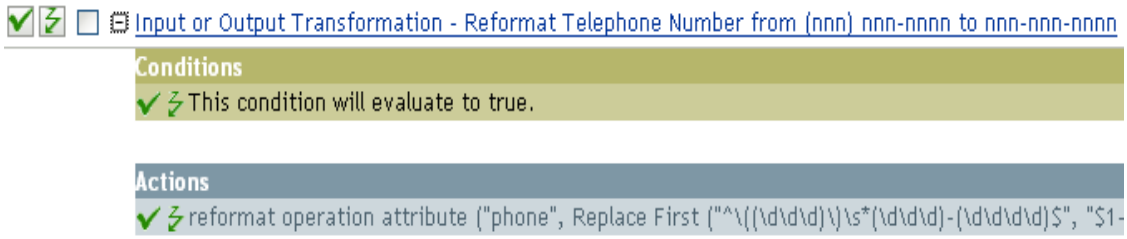
Specify the syntax of the new attribute value.

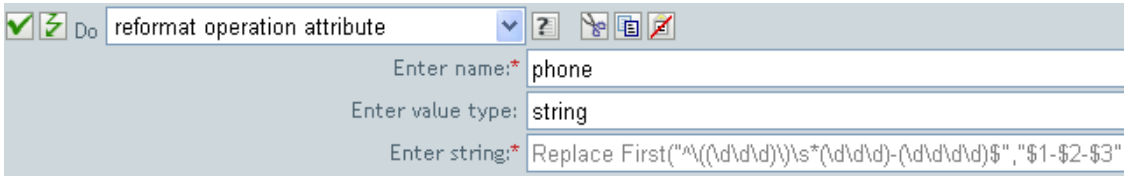
Value

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.

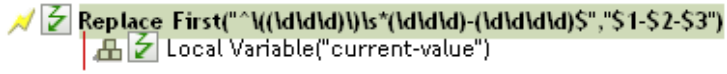
Example

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" on page 57. To view the policy in XML, see predef_transformation_reformat_telephone1.xml (../samples/predef_transformation_reformat_telephone1.xml).





The action reformat operation attribute changes the format of the telephone number. The rule uses the Argument Builder and regular expressions to change how the information is displayed.



Remove Association

Sends a remove association command to the Identity Vault.

Fields

Mode

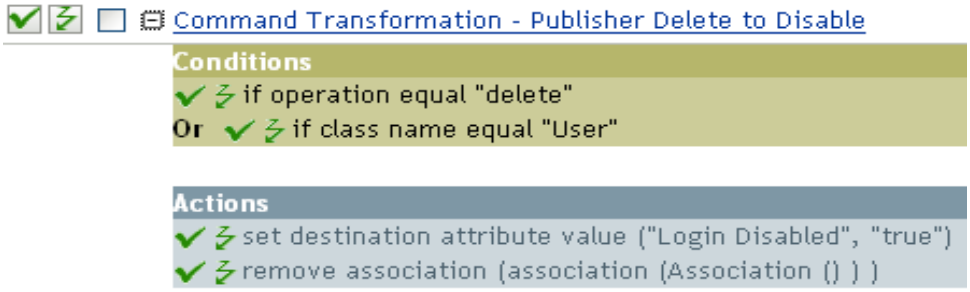
Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

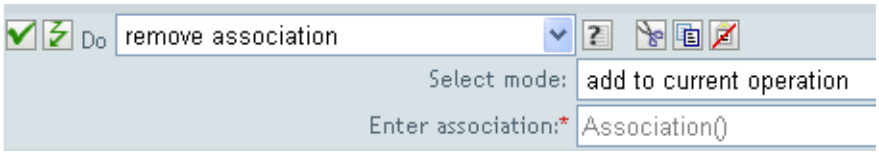
Association

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” on page 50. To view the policy in XML, see predef_command_delete_to_disable.xml (../samples/predef_command_delete_to_disable.xml).





When a delete operation occurs for a User object, value of the Login Disabled attribute is set to true and the association is removed from the object. The association is removed because the associated object in the connected application no longer exists.

Remove Destination Attribute Value

Removes an attribute value from an object in the destination data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 207).

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 207).

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.

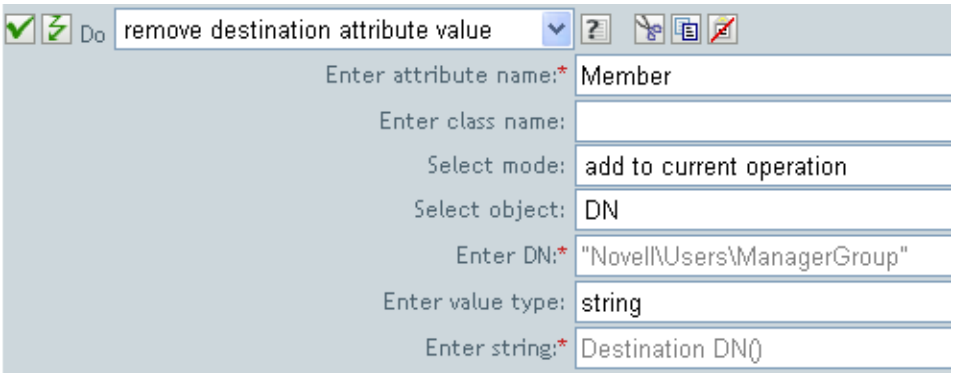
Value Type

Specify the syntax of the new attribute value.

String

Specify the value of the new attribute.

Example



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 207).

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 207).

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

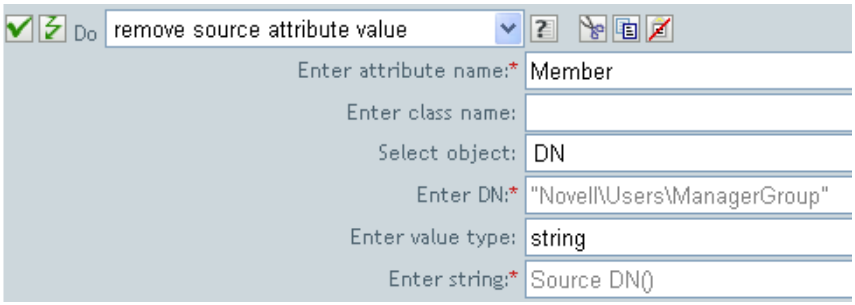
Value Type

Specify the syntax of the attribute value to be removed.

String

Specify the attribute value to be removed.

Example



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.

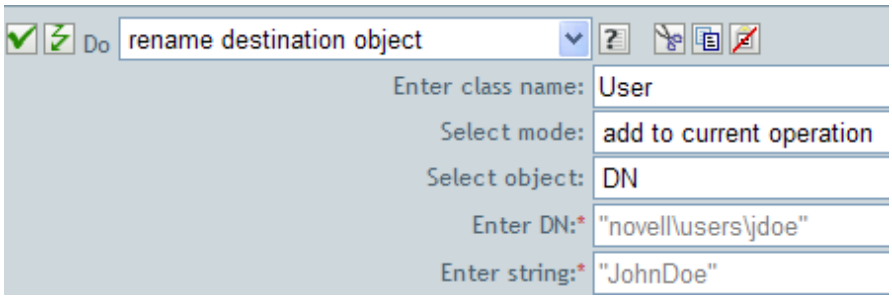
Object

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



Rename Operation Attribute

Renames all occurrences of an attribute within the current operation.

Fields

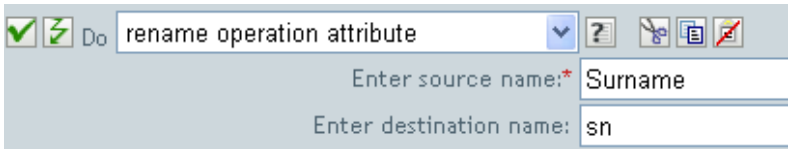
Source Name

Specify the original attribute name. Supports variable expansion. For more information, see Variable Expansion (page 207).

Destination Name

Specify the new attribute name. Supports variable expansion. For more information, see Variable Expansion (page 207).

Example



Rename Source Object

Renames an object in the source data store.

Fields

Class Name

(Optional) Specify the class name of the object to rename in the source data store.

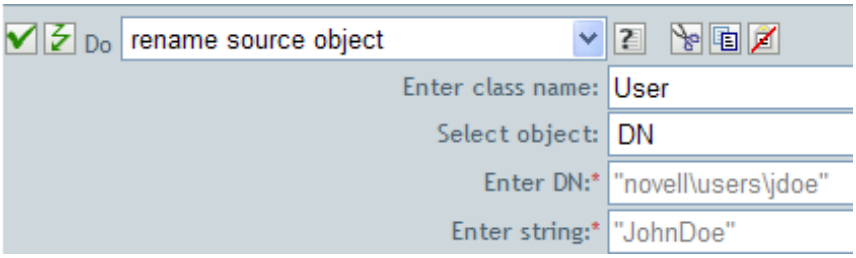
Select Object

Select the target object. This object can be the current object, or it can be specified by a DN or an association.

String

Specify the new name of the object.

Example



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 207).

Server

Specify the SMTP server name. Supports variable expansion. For more information, see Variable Expansion (page 207).

Message Type

Select the e-mail message type.

Password

(Optional) Specify the SMTP server account password.

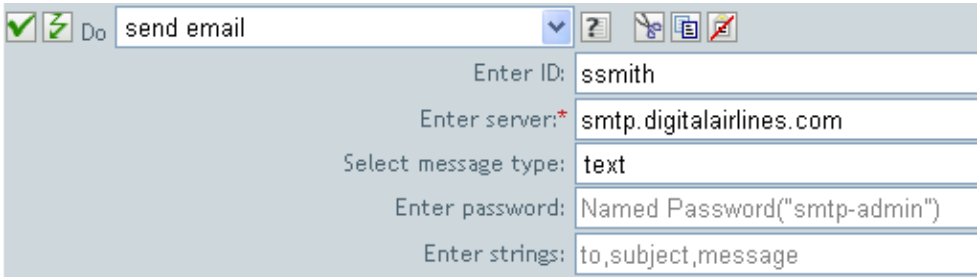
IMPORTANT: You can store the SMTP server account password as a Named Password on the driver object. This allows the password to be encrypted; otherwise you enter the password and it is stored in clear text. For more information on Named Passwords, see "Using Named Passwords" in the *Novell Identity Manager 3.6 Administration Guide*.

Strings

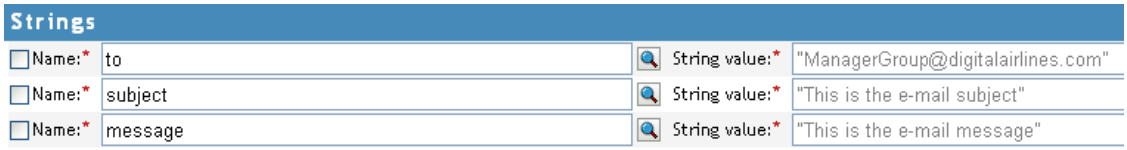
Specify the values containing the various e-mail addresses, subject, and message. The following table lists valid named string arguments:

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



The following is an example of the Named String Builder being used to provide the strings argument:



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 207).

Template DN

Specify the slash form DN of the e-mail template object. Supports variable expansion. For more information, see Variable Expansion (page 207).

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 3.6 Administration Guide*.

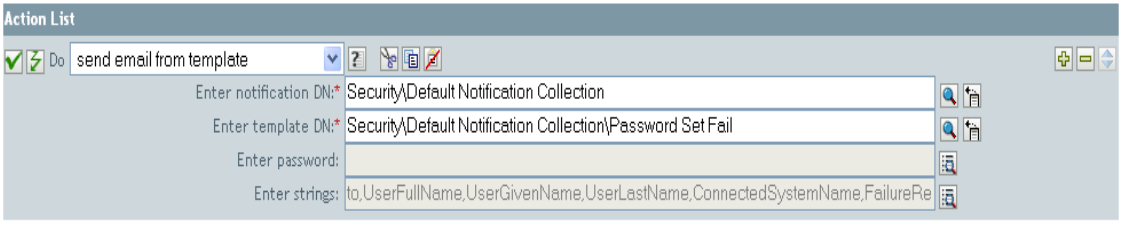
Strings

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. If you want to use HTML tags to format the strings, use the HTML tags within the `<use-html></use-html>` tags. The value of <arg-string> tag attribute is interpretted as HTML, if it is enclosed within `<use-html></use-html>` tags.

Example



The following is an example of the Named String Builder, used to provide the strings argument:

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 207).

Write Back

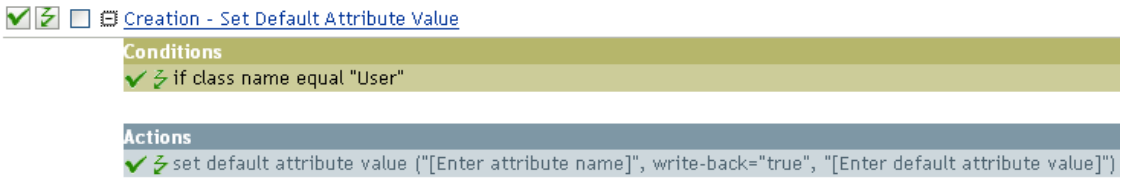
Select whether or not to also write back the default values to the source data store.

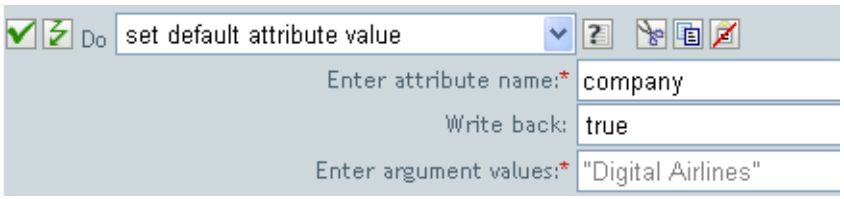
Values

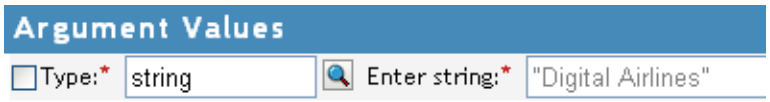
Specify the default values of the attribute.

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" on page 53. To view the policy in XML, see predef_creation_set_default_attribute_value.xml (../samples/predef_creation_set_default_attribute_value.xml).







To build the value, the Argument Value List Builder is launched. See "Argument Value List Builder" on page 31 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 207).

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 207).

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

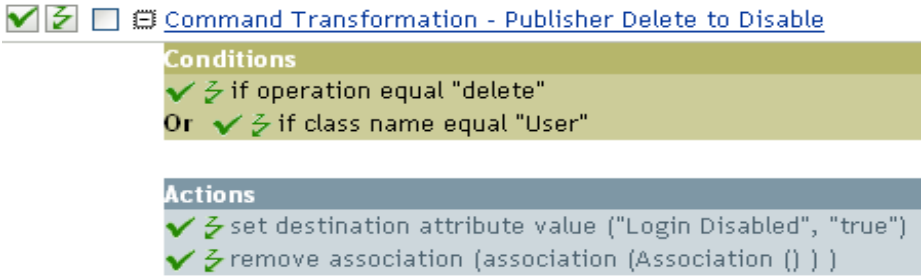
Select the syntax of the attribute value to set.

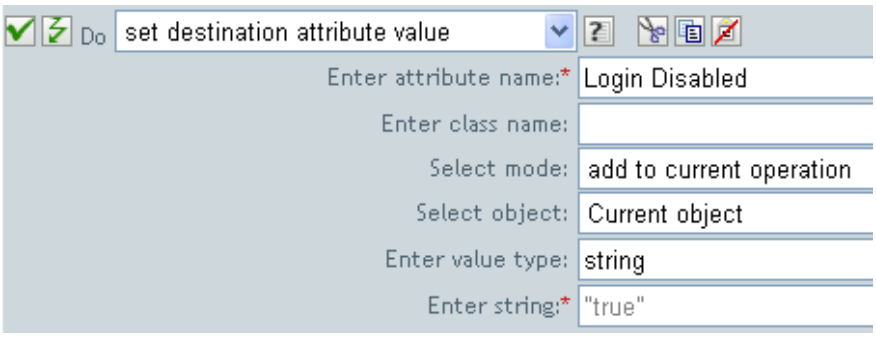
String

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" on page 50. To view the policy in XML, see predef_command_delete_to_disable.xml (../samples/predef_command_delete_to_disable.xml).





The rule sets the value for the attribute of Login Disabled to true. The rule uses the Argument Builder to add the text of true as the value of the attribute. See “Argument Builder” on page 26 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

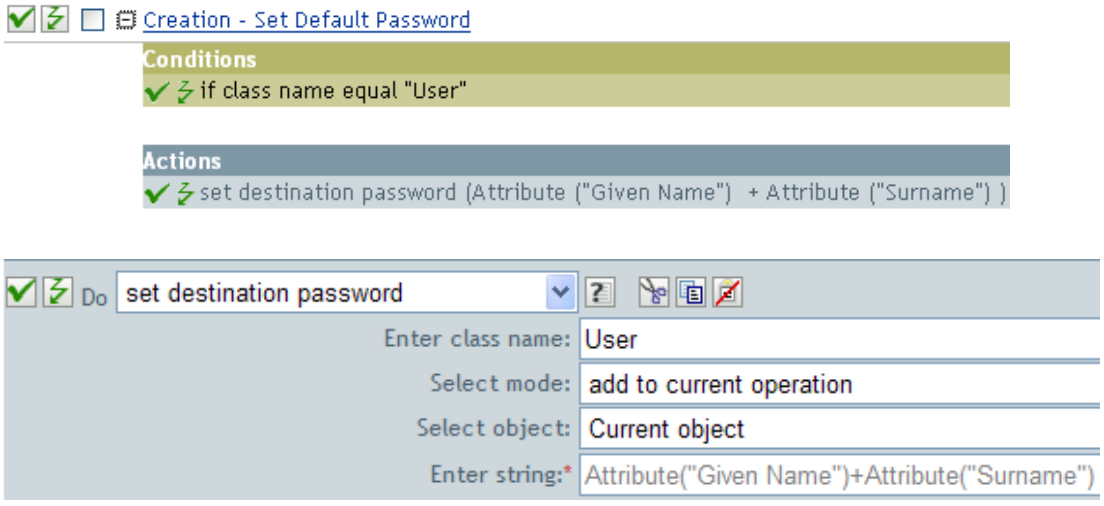
Select the target object. This object can be the current object, or can be specified by an DN or an association.

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" on page 54.



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 207).

Variable Scope

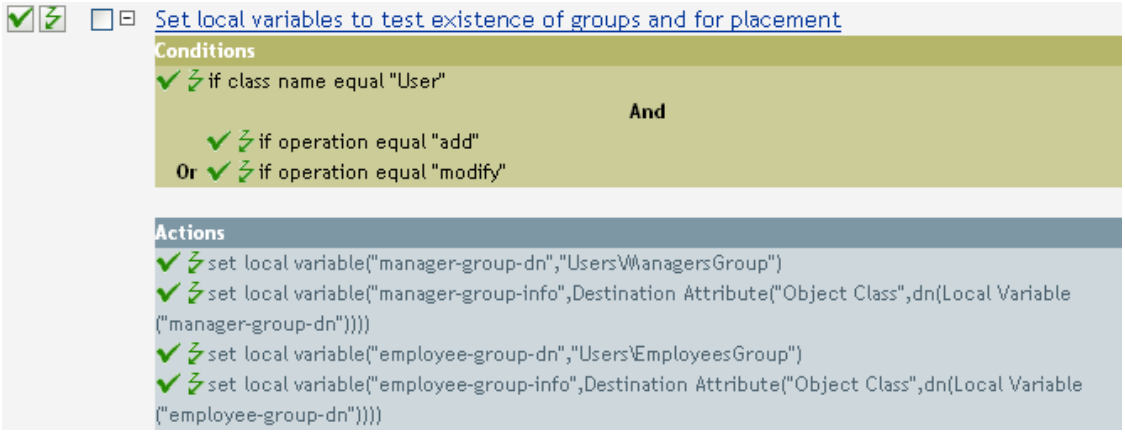
Select the scope of the local variable. This can be set to the driver or to the policy. Supports variable expansion. For more information, see Variable Expansion (page 207).

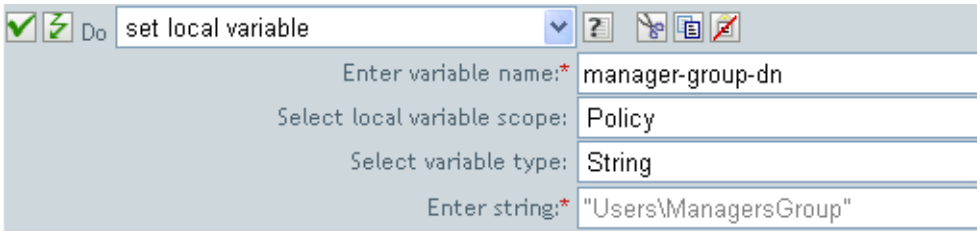
Variable Type

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 003-AddCreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).





The local variable is set to the value that is in the User object's destination attribute of Object Class plus the Local Variable of manager-group-info. The Argument Builder is used to construct the local variable. See "Argument Builder" on page 26 for more information.

Set Operation Association

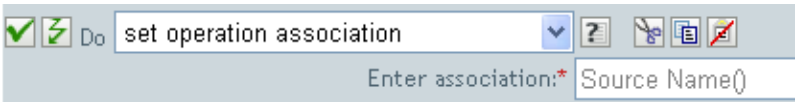
Sets the association value for the current operation.

Fields

Association

Provide the new association value.

Example



Set Operation Class Name

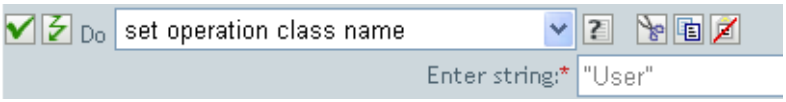
Sets the object class name for the current operation.

Fields

String

Specify the new class name.

Example



Set Operation Destination DN

Sets the destination DN for the current operation.

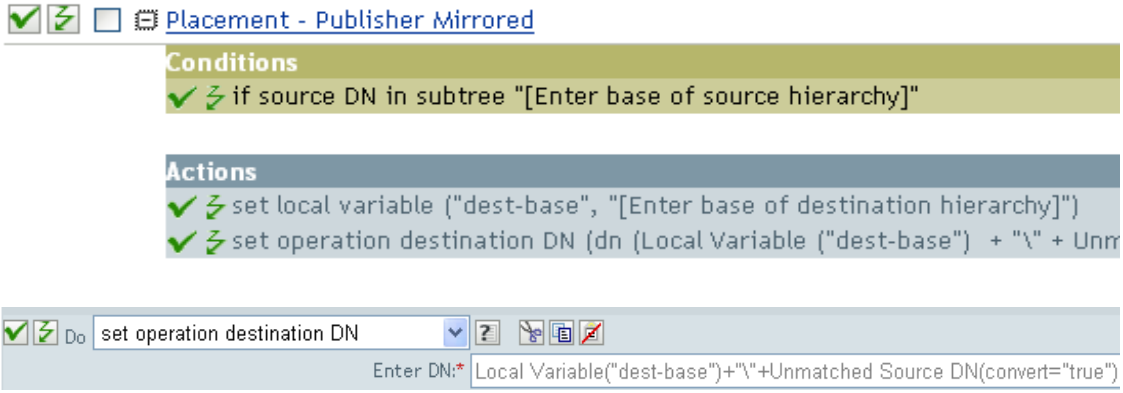
Fields

DN

Specify the new destination DN.

Example

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” on page 63. To view the policy in XML, see predef_place_pub_mirrored.xml (../samples/predef_place_pub_mirrored.xml).



The rule sets the operation destination DN to be the local variable of the destination base location plus the source DN.

Set Operation Property

Sets an operation property. An operation property is a named value that is stored within an operation. It is typically used to supply additional context that might be needed by the policy that handles the results of an operation.

Fields

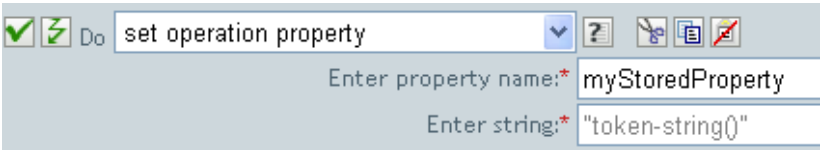
Property Name

Specify the name of the operation property. Supports variable expansion. For more information, see Variable Expansion (page 207).

String

Specify the name of the string.

Example



Set Operation Source DN

Sets the source DN for the current operation.

Fields

DN

Specify the new source DN.

Example

Set Operation Template DN

Sets the template DN for the current operation to the specified value. This action is only valid when the current operation is add.

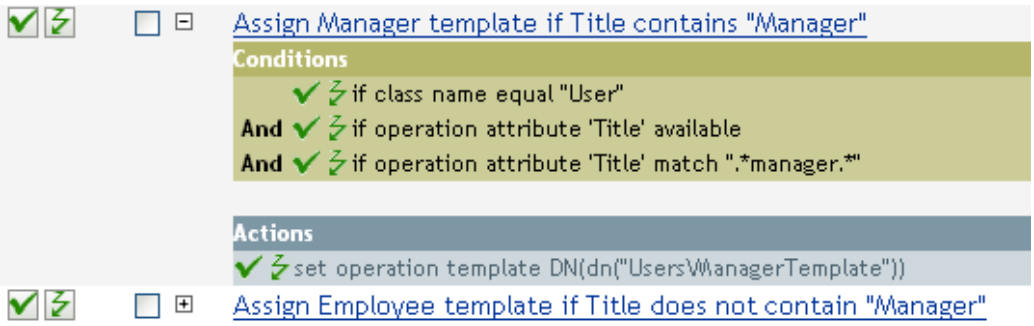
Fields

DN

Specify the template DN.

Example

The example applies the Manager template if the Title attribute contains the word Manager. The name of the policy is Policy: Assign Template to User Based on Tile, and it is available for download from the Novell Support Web site. For more information, see "Downloading Identity Manager Policies" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 003-Create-AssignTemplateByTitle.xml (../samples/003-Create-AssignTemplateByTitle.xml).



The template Manager Template is applied to any User object the has the attribute of Title *available* and contains the word Manager somewhere in the title. The policy uses regular expressions to find all possible matches.

Set Source Attribute Value

Adds a value to an attribute on an object in the source data store, and removes all other values for that attribute.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 207).

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 207).

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

Value Type

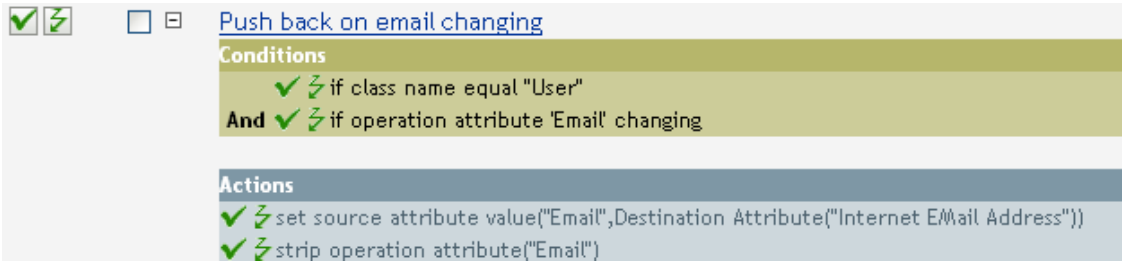
Select the syntax of the attribute value.

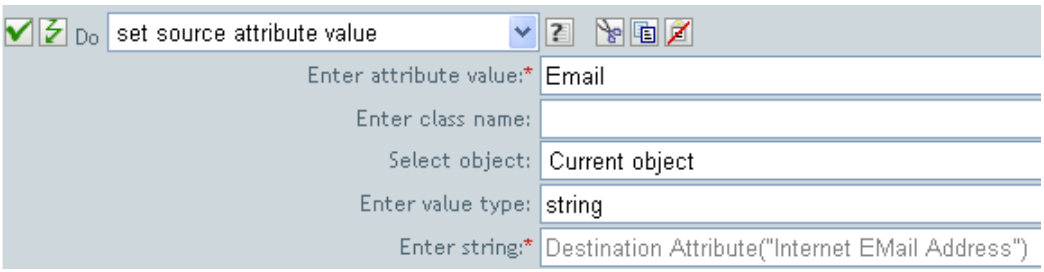
Value

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 001-Input_PushBackOnEmail (../samples/001-Input-PushBackOnEmail.xml).





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.

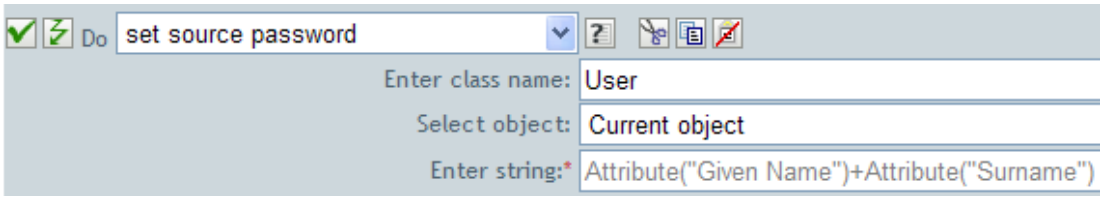
Object

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



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 for Identity Manager 3.6*.

Fields

Credential Store Object DN

Specify the DN of the repository object. Supports variable expansion. For more information, see Variable Expansion (page 207).

Target User DN

Specify the DN of the target users.

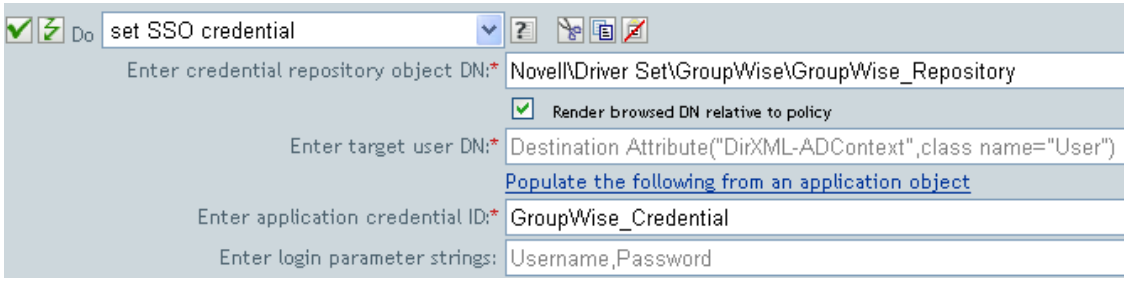
Application Credential ID

Specify the application credential that is stored in the application object. Supports variable expansion. For more information, see Variable Expansion (page 207).

Login Parameter Strings

Specify each login parameter for the application. The login parameters are the authentication keys stored in the application object.

Example



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 for Identity Manager 3.6*.

Fields

Credential Store Object DN

Specify the DN of the repository object. Supports variable expansion. For more information, see Variable Expansion (page 207).

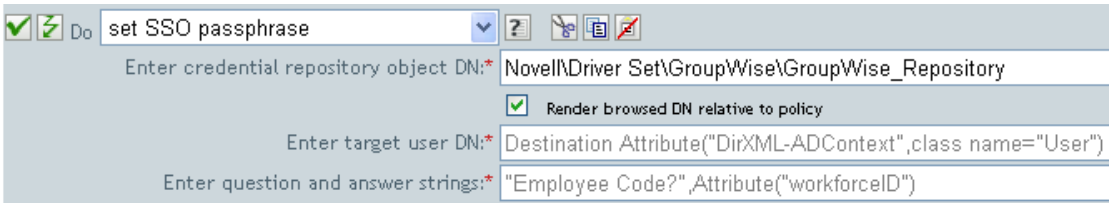
Target User DN

Specify the DN of the target users.

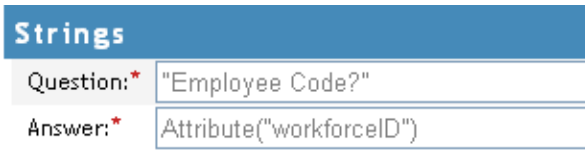
Question and Answer Strings

Specify the SecureLogin passphrase question and answer.

Example



The SecureLogin passphrase question and answer are stored as strings in the policy. Click the *Edit these strings* icon to launch the string builder. Specify the passphrase question and 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. Supports variable expansion. For more information, see Variable Expansion (page 207).

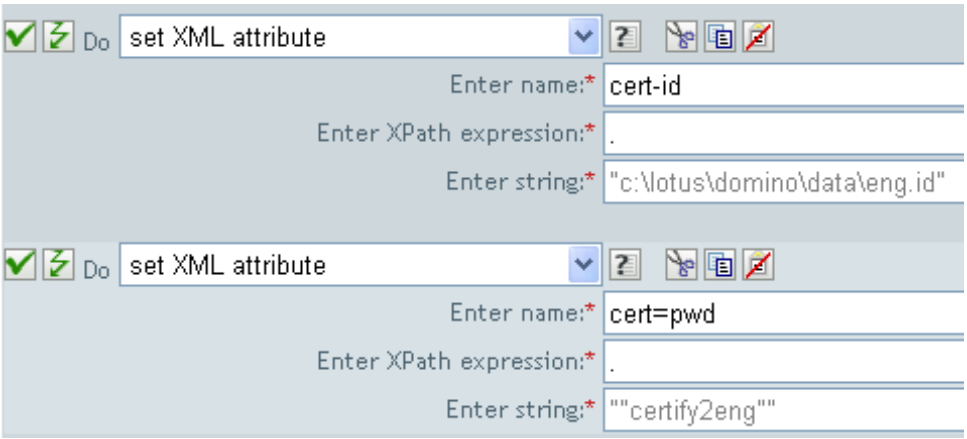
XPath Expression

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



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 207).

Message

Provide the status message using the Argument Builder.

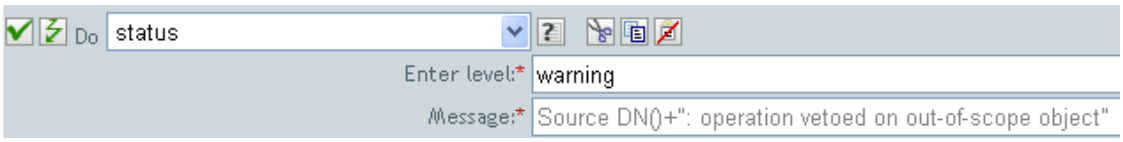
Remarks

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



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 207).

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 207).

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 207).

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

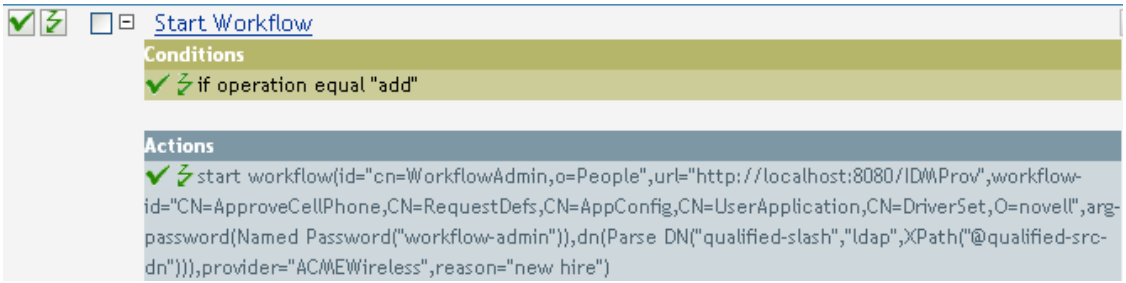
Specify the DN of the recipient of the workflow in LDAP format.

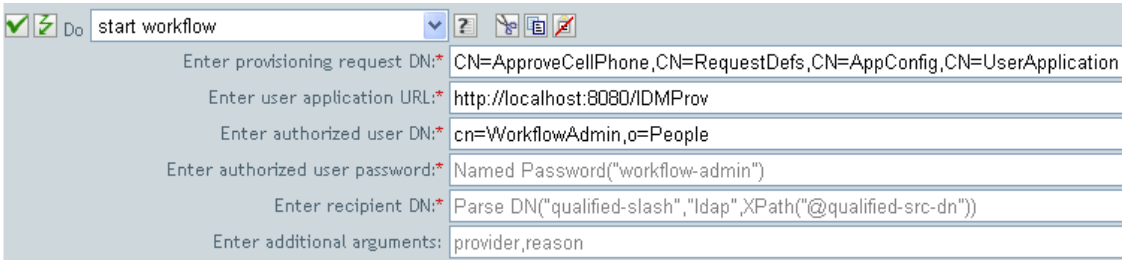
Additional Arguments

Specify the arguments for the workflow. The arguments are different for each workflow.

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).





Strip Operation Attribute

Strips all occurrences of an attribute from the current operation.

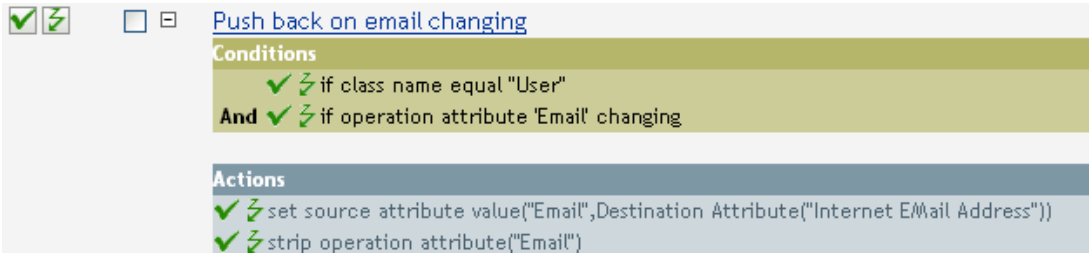
Fields

Name

Specify the name of the attribute to be stripped. Supports variable expansion. For more information, see Variable Expansion (page 207).

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 001-Input-PushBackOnEmail.xml (../samples/001-Input-PushBackOnEmail.xml).



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

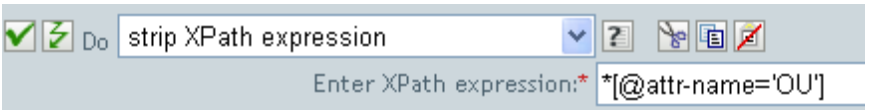
XPath Expression

Specify the XPath 1.0 expression that returns a node set containing the nodes to be stripped.

Remarks

For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *Understanding Policies for Identity Manager 3.6*.

Example



Trace Message

Sends a message to DSTRACE.

Fields

Level

Specify the trace level of the message. The default level is 0. The message only appears if the specified trace level is less than or equal to the trace level configured in the driver.

For information on how to set the trace level on the driver, see "Viewing Identity Manager Processes" in the *Novell Identity Manager 3.6 Administration Guide*.

Color

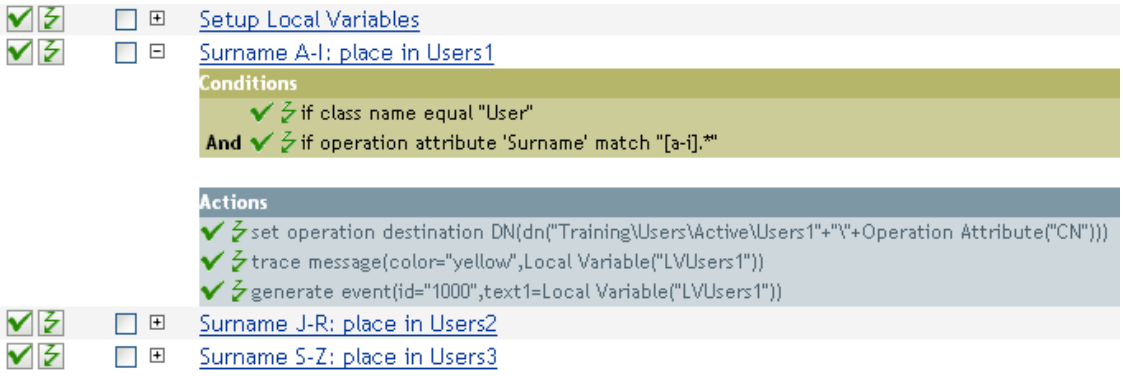
Select the color of the trace message.

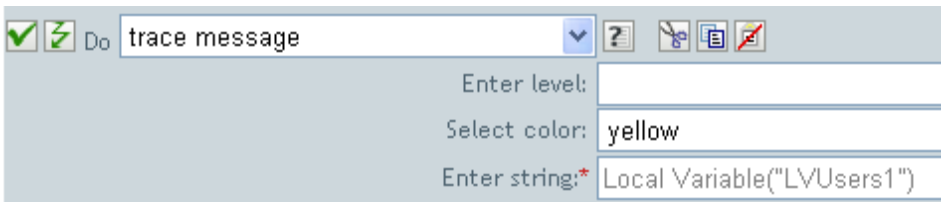
String

Specify the value of the trace message.

Example

The example has four rules that implement a Placement policy for User objects based on the first character of the Surname attribute. It generates both a trace message and a custom 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 "Downloading Identity Manager Policies" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 001-Placement-BySurname.xml (../samples/001-Placement-BySurname.xml).





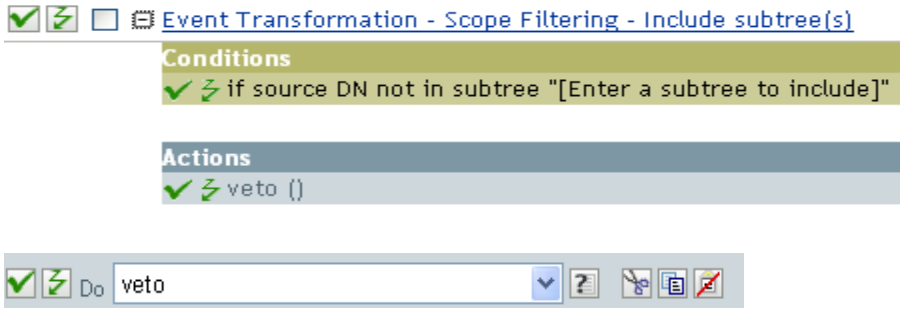
The action sends a trace message to DSTRACE. The contents of the local variable is LVUsers1 and it shows up in yellow in DSTRACE.

Veto

Vetoes the current operation.

Example

The example excludes all events that come from the specified subtree. The rule is from the predefined rules that come with Identity Manager. For more information, see “Event Transformation - Scope Filtering - Exclude Subtrees” on page 56 from the predefined rules. To view the policy in XML, see predef_transformation_filter_exclude_subtress.xml (../samples/predef_transformation_filter_exclude_subtrees.xml).



The action vetoes all events that come from the specified subtree.

Veto If Operation Attribute Not Available

Conditionally cancels the current operation and ends processing of the current policy, based on the availability of an attribute in the current operation.

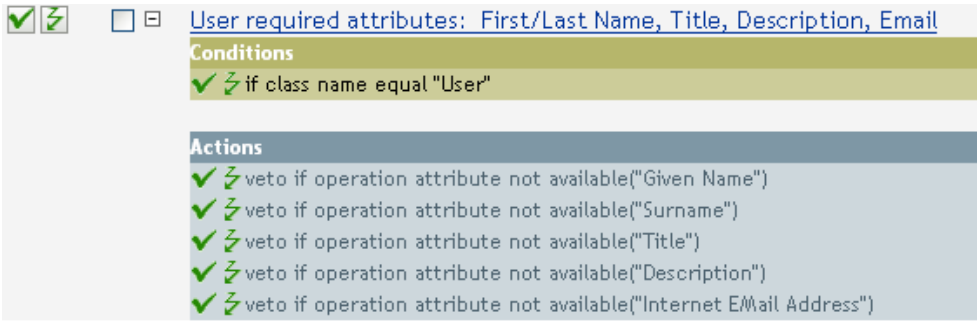
Fields

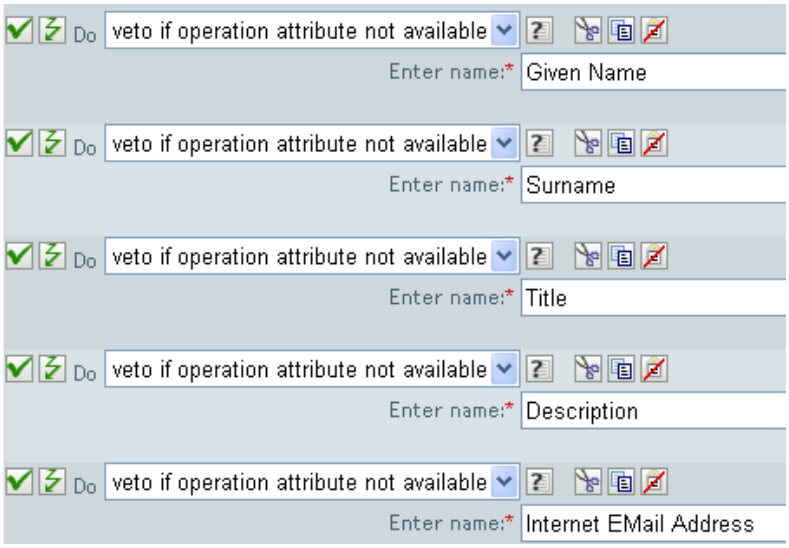
Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 207).

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 001-Create-RequiredAttrs.xml (../samples/001-Create-RequiredAttrs.xml).





The actions vetoes the operation if the attributes of Given Name, Surname, Title, Description, and Internet Email Address are not available.

While

Causes the specified actions to be repeated while the specified conditions evaluate to True.

Fields

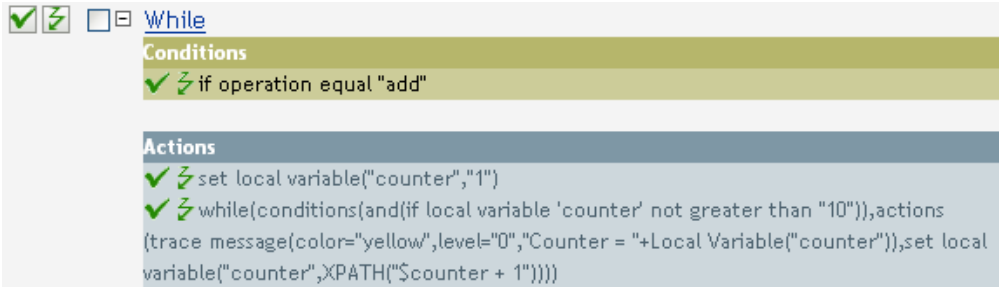
Conditions

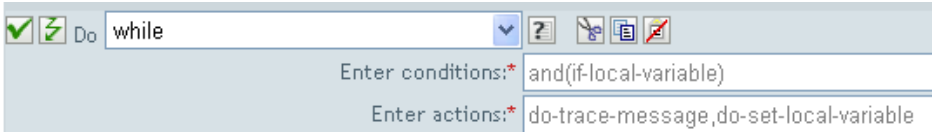
Specify the condition to be evaluated.

Actions

Specify the actions to be repeated if the conditions evaluate to True.

Example





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 from $<variable-name>$ is replaced with the value of the local or global variable with the five name. $<variable-name>$ must be a legal variable name. For more information on what is a legal XML name, see W3C Extensible Markup Language (http://www.w3.org/TR/2006/REC-xml11-20060816/#sec-suggested-names).

Noun Tokens

11

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 246).

Remarks

If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that entitlement. If it is used in a context where a string is expected, the token expands to the string value found.

Example

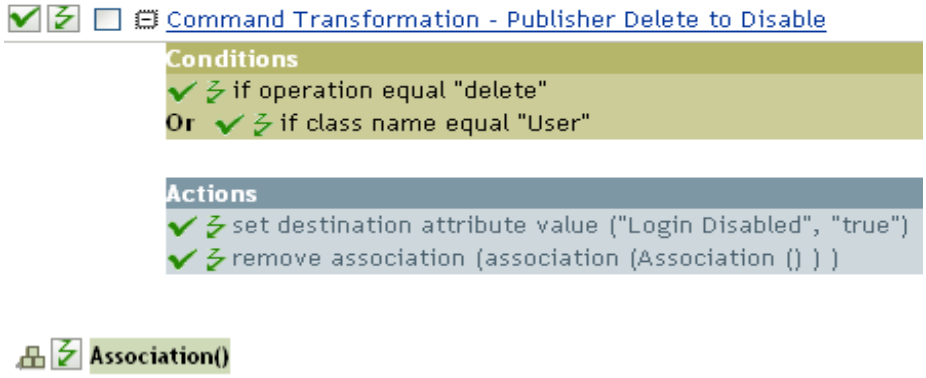
Association

Expands to the association value from the current operation.

Example

The example is from the predefined rules that come with Identity Manager. For more information on the predefined rule, see “Command Transformation - Publisher Delete to Disable” on page 50.

The action of Remove Association uses the Association token to retrieve the value from the current operation. The rule removes the association from the User object so that any new events coming through do not affect the User object. To view the policy in XML, see predef_command_delete_to_disable.xml (../samples/predef_command_delete_to_disable.xml).



Attribute

Expands to the value of an attribute from the current object in the current operation and in the source data store. It can be logically thought of as the union of the operation attribute token and the source attribute token. It does not include the removed values from a modify operation.

Fields

Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 246).

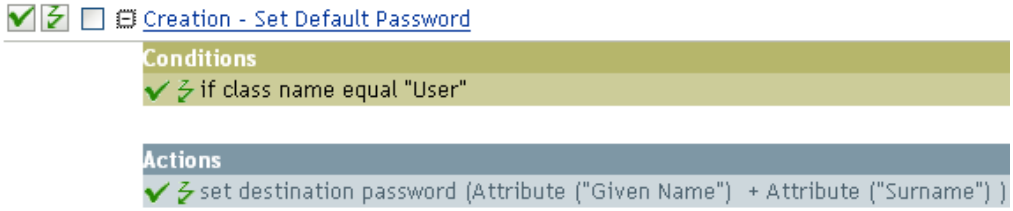
Remarks

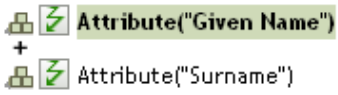
If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that attribute. If it is used in a context where a string is expected, the token expands to the string value found.

Example

The example is from the predefined rules that come with Identity Manager. For more information, see “Creation - Set Default Password” on page 54.

The action of Set Destination Password uses the attribute token to create the password. The password is made up of the Given Name attribute and the Surname attribute. When you are in the Argument Builder Editor, you browse and select the attribute you want to use. To view the policy in XML, see predef_creation_set_default_password.xml (../samples/predef_creation_set_default_password.xml).





Character

Expands to a character specified by a Unicode* code point.

Remarks

For a listing of Unicode values and characters, see Unicode Code Charts (http://www.unicode.org/charts/).

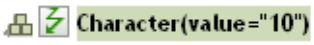
Fields

Character Value

The Unicode code point of the character. Supports variable expansion. For more information, see Variable Expansion (page 246).

A hexadecimal number can be specified if it is prefixed with `0x`, as in C-based programming languages.

Example



Class Name

Expands to the object class name from the current operation.

Example

Destination Attribute

Expands to the specified attribute value an object.

Fields

Name

Name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 246).

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 246).

Select Object

Select Current Object, DN, or Association.

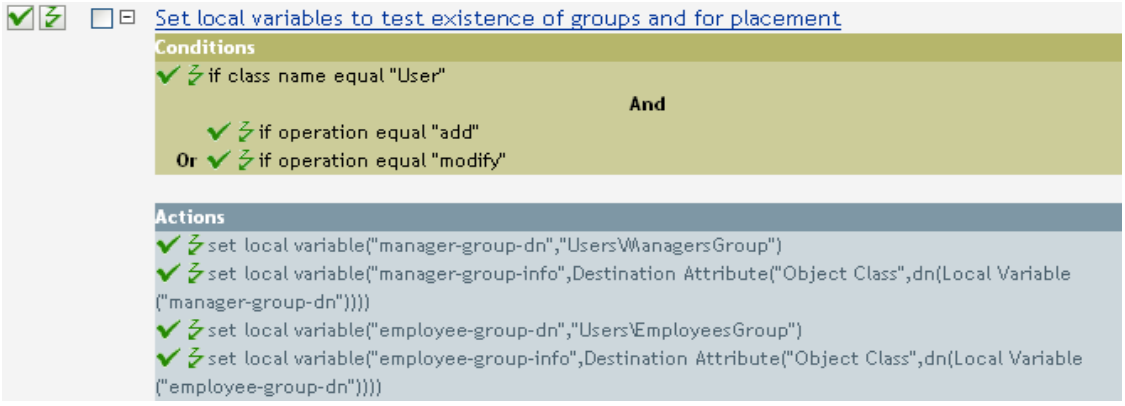
Remarks

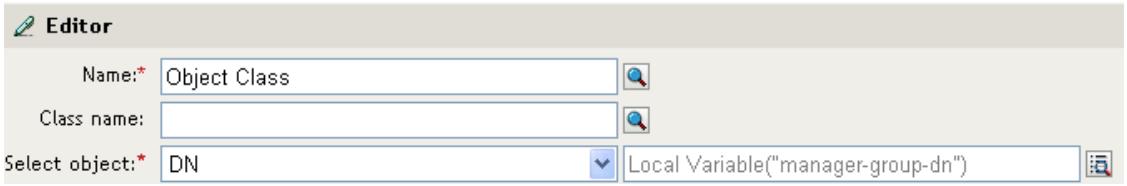
If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that attribute. If it is used in a context where a string is expected, the token expands to the string value found.

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 003-Command-AddCreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).

The policy creates the Destination Attribute with the Argument Builder. The action of Set Local Variable contains the Destination Attribute token.





You build the Destination Attribute through the Editor. In this example, the attribute of Object Class is set. 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

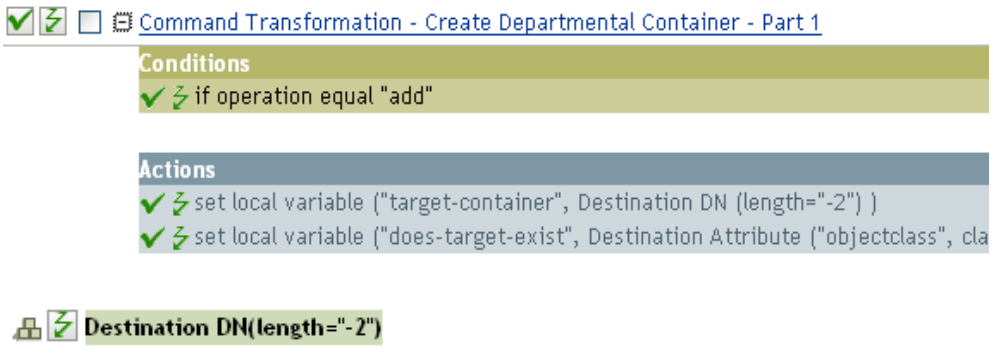
Specify the number of RDN segments to include. Negative numbers are interpreted as (total # of segments + length) + 1. For example, for a DN with 5 segments a length of -1 = (5 + (-1)) + 1 = 5, -2 = (5 + (-2)) + 1 = 4, etc.

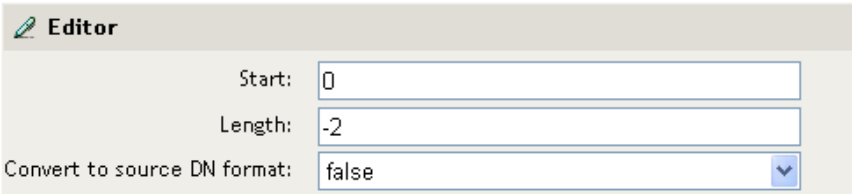
Remarks

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" on page 48. To view the policy in XML, see predef_command_create_dept_container1.xml (../samples/predef_command_create_dept_container1.xml).




Editor
Start:
0
Length:
-2
Convert to source DN format:
false

Destination Name

Expands to the unqualified Relative Distinguished Name (RDN) of the destination DN specified in the current operation.

Example

Document

Reads the XML document pointed to by the URI and returns the document node in a node set. The URI can be relative to the URI of the including policy. With any error, the result is an empty node set.

Fields

XML Document URI

Specify the XML document URI.

Example

Entitlement

Expands to the values of a granted entitlement from the current object.

Fields

Name

Name of the entitlement. Supports variable expansion. For more information, see Variable Expansion (page 246).

Remarks

If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that entitlement. If it is used in a context where a string is expected, the token expands to the string value found.

Example

Generate Password

Generates a random password that conforms to the 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 246).

Render browsed DN relative to policy

Select whether the DN of the password policy is relative to the policy being created.

Example

Global Configuration Value

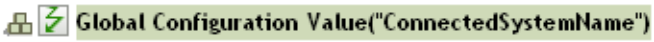
Expands to the value of a global configuration variable.

Fields

Name

Name of the global configuration value. Supports variable expansion. For more information, see Variable Expansion (page 246).

Example



Local Variable

Expands to the value of a local variable.

Fields

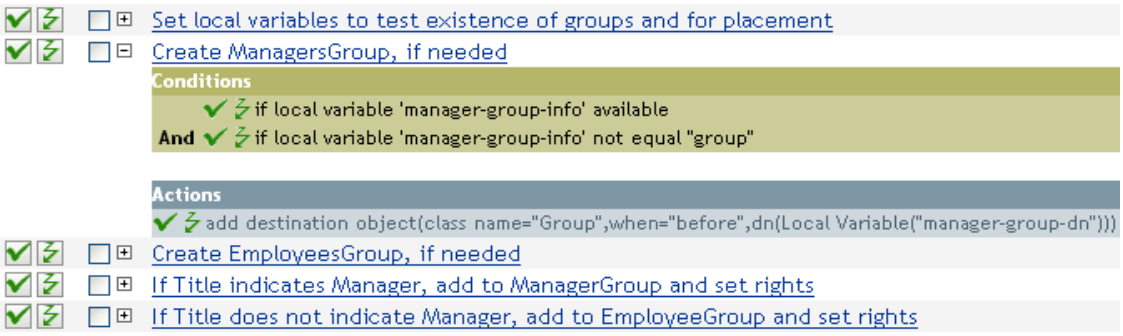
Name

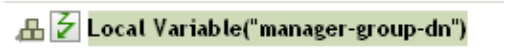
Specify the name of the local variable. Supports variable expansion. For more information, see Variable Expansion (page 246).

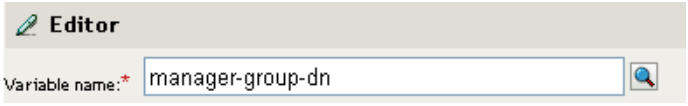
Example

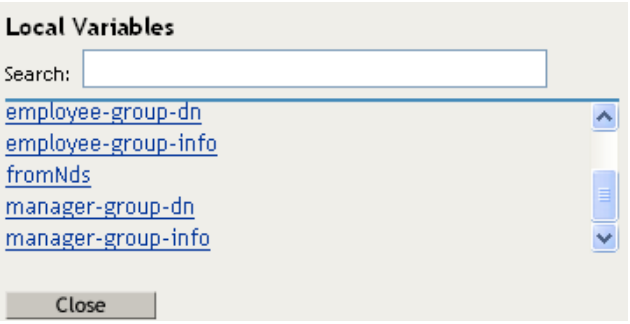
The example is from the Govern Groups for User Based on Title policy, which is available for download from the Novell Support Web site. For more information, see "Downloading Identity Manager Policies" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 003-Command-AddCreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).

The action Add Destination Object uses the Local Variable token.









The Local Variable can only be used if the action Set Local Variable has been used previously in the policy. It sets the value that is stored in the Local Variable. In the Editor, you click the browse icon and all of the local variables that have been defined are listed. Select the correct local variable.

The value of the local variable is group-manager-dn. In the example, the Set Local Variable action defined group-manager-dn as DN of the manager's group Users\ManagersGroup.

Named Password

Expands to the named password from the driver.

Fields

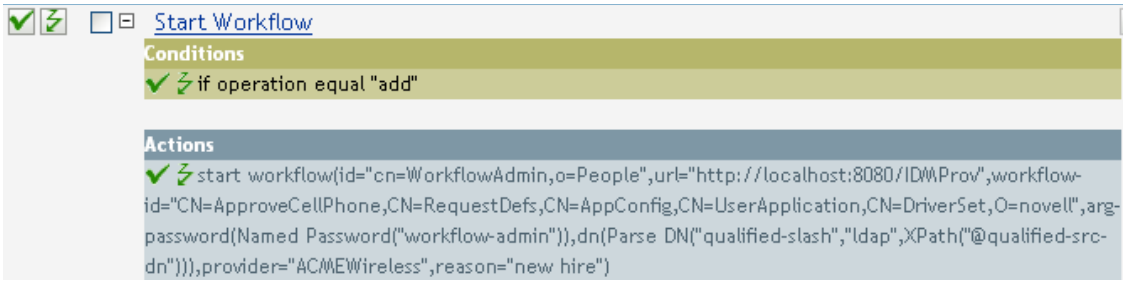
Name

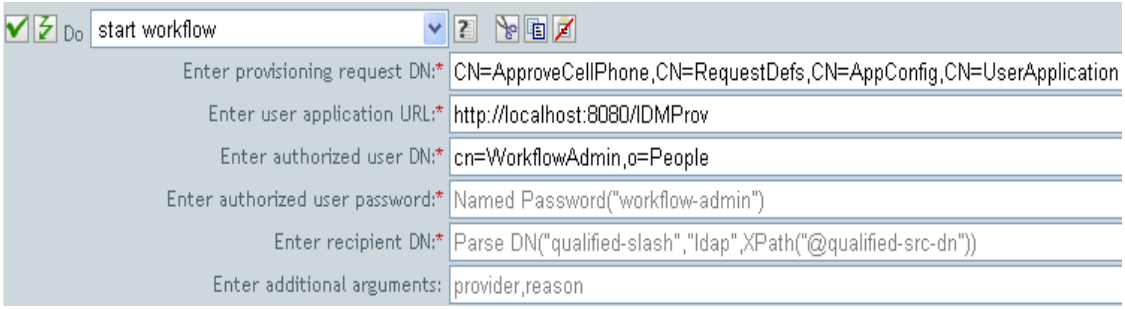
Name of the password. Supports variable expansion. For more information, see Variable Expansion (page 246).

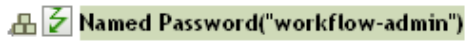
Example

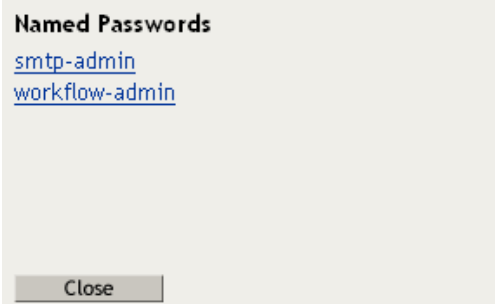
The Named Password noun token can only be used if a Named Password has been set on the driver object. The Named Password is used to save a password in an encrypted form. Sometimes it is required to provide a password to allow an action to function. If you enter the password as clear text, it is a security risk.

The example uses the Start Workflow action. It requires that the password for the workflow administrator be entered. To view the policy in XML, see start_workflow.xml (../samples/start_workflow.xml).








Named Passwords
smtp-admin
workflow-admin
Close

Operation

Expands to the name of the current operation.

Example

Operation Attribute

Expands to the value of an attribute in the current operation. The operation can be an <add-attr>, <add-value>, or <attr>. If this token is evaluated in a context where a node-set result is expected then all the available values are returned as nodes in a node-set. Otherwise the first available value is returned as a string.

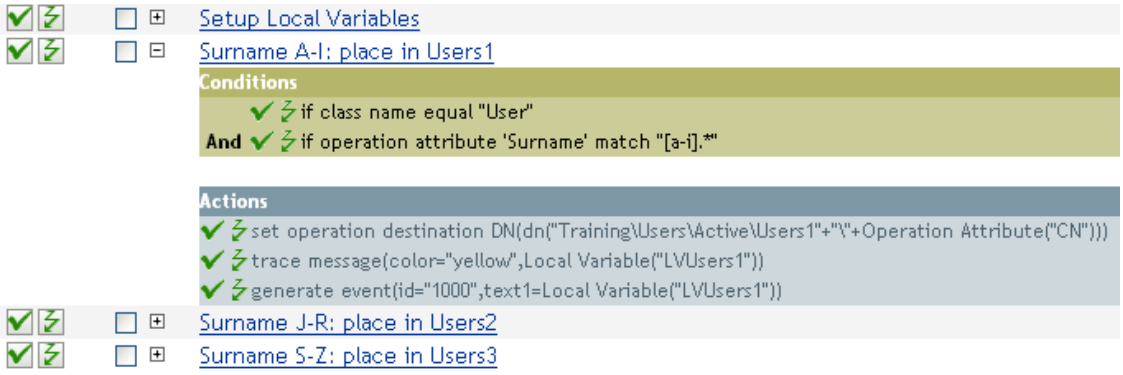
Fields

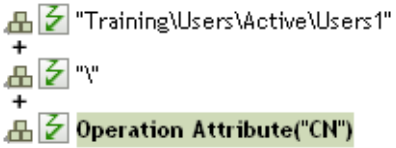
Name

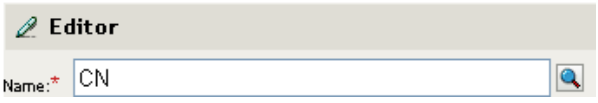
Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 246).

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 001-Placement-BySurname.xml (../samples/001-Placement-BySurname.xml).







The action Set Operation Destination DN contains the Operation Attribute token. The Operation Attribute token sets the Destination DN to the CN attribute. The rule takes the context of Training\Users\Active\Users and adds a \ plus the value of the CN attribute.

Operation Property

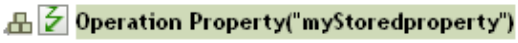
Expands to the value of the specified operation property on the current operation.

Fields

Name

Specify the name of the operation property. Supports variable expansion. For more information, see Variable Expansion (page 246).

Example



Password

Expands to the password specified in the current operation.

Example

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 246).

Select Object

Specify the base of the query. It can be the current object, DN, or an association.

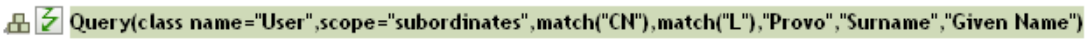
Match Attributes

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



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 246).

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 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 246).

Remarks

If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that entitlement. If it is used in a context where a string is expected, the token expands to the string value found.

Example

Resolve

Resolves the DN to an association key, or the association key to a DN in the specified data store.

Fields

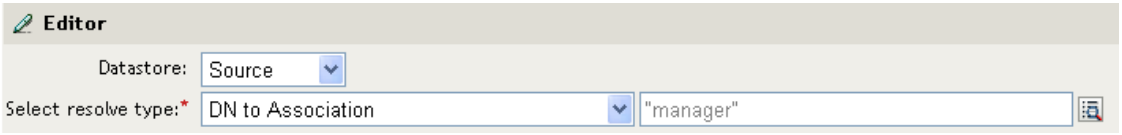
Datastore

Select the destination or source 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



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 246).

Name

Name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 246).

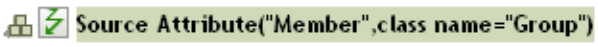
Object

Select the source object. This object can be the current object, or can be specified by a DN or an association.

Remarks

If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that attribute. If it is used in a context where a string is expected, the token expands to the string value found.

Example



Source DN

Expands to the source DN from the current operation.

Fields

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

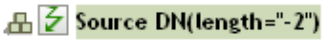
Length

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 Name

Expands to the unqualified relative distinguished name (RDN) of the source DN specified in the current operation.

Example

Time

Expands to the current date/time into the format, language, and time zone specified.

Fields

Format

Specify the date/time format. Select a named time format or specify a custom format pattern. Supports variable expansion. For more information, see Variable Expansion (page 246).

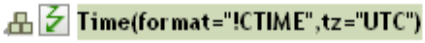
Language

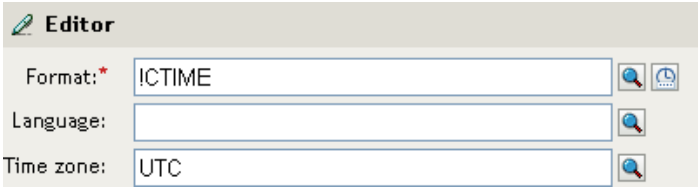
Specify the language. (It defaults to the current system language.) Supports variable expansion. For more information, see Variable Expansion (page 246).

Time zone

Specify the time zone. (It defaults to the current system time zone.) Supports variable expansion. For more information, see Variable Expansion (page 246).

Example





Text

Expands to the text.

Fields

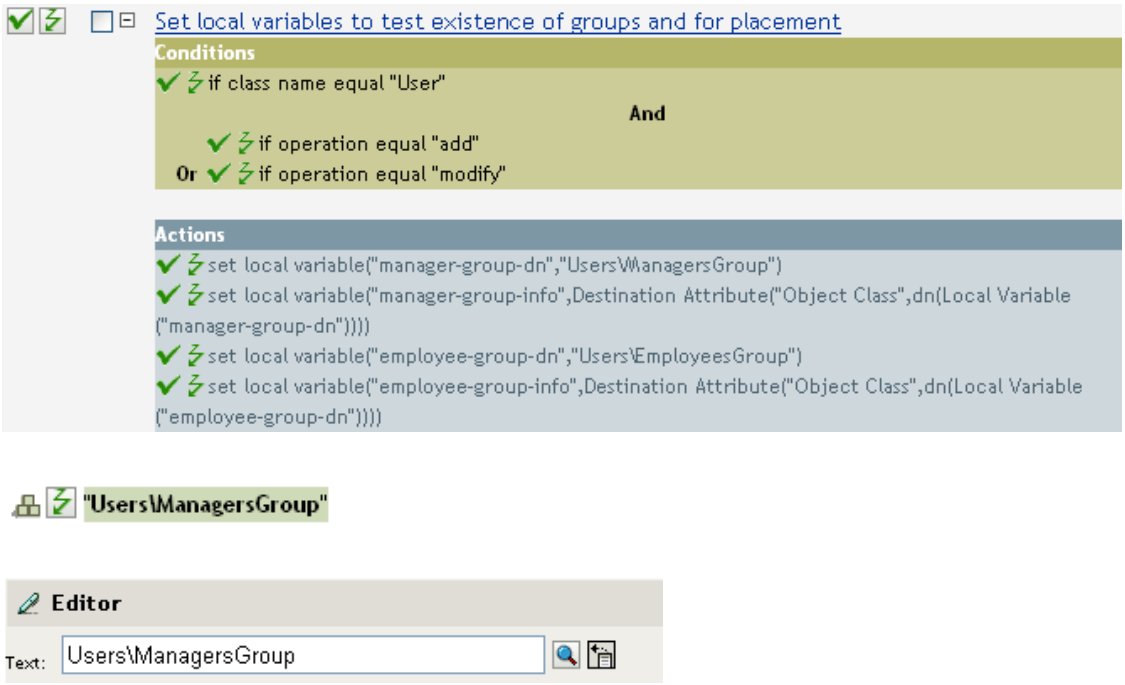
Text

Specify the text. Supports variable expansion. For more information, see Variable Expansion (page 246).

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" in *Understanding Policies for Identity Manager 3.6*.To view the policy in XML, see 003-Command-AddCreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).

The Text token is used in the action Set Location Variable to define the DN of the manager's group. The Text token can contain objects or plain text.



The Text token contains the DN for the manager's group. You can browse to the object you want 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 use a counter
- Never use a counter
- After all patterns failed without

Counters Pattern

Select which pattern to use the counter with. The options are:

- Only with first pattern
- Only with last pattern
- Use with all patterns

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.

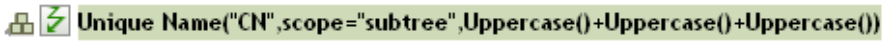
As soon as a proposed name is determined to be unique, the testing of names is stopped and the unique name is returned.

The order of proposed names is tested as follows:

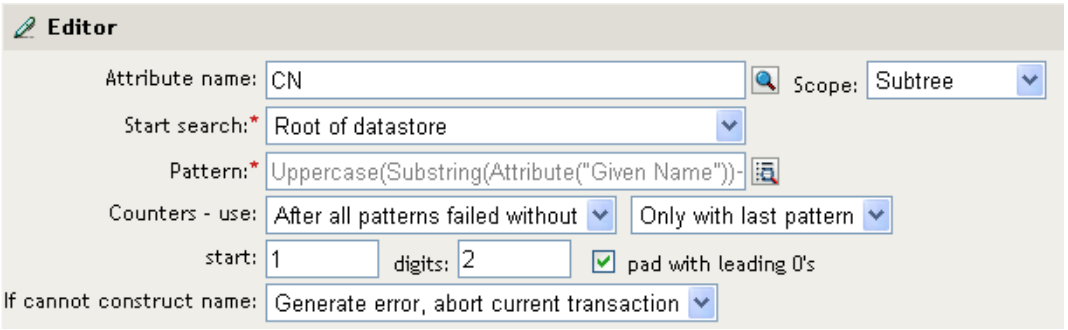
- Each pattern is tested in the order specified. If counter-use=“always” and the pattern is one of the patterns indicated by the counter-pattern then the pattern is tested with a counter, otherwise it is tested without a counter.
- If no unique name has been found after the patterns have been exhausted 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.

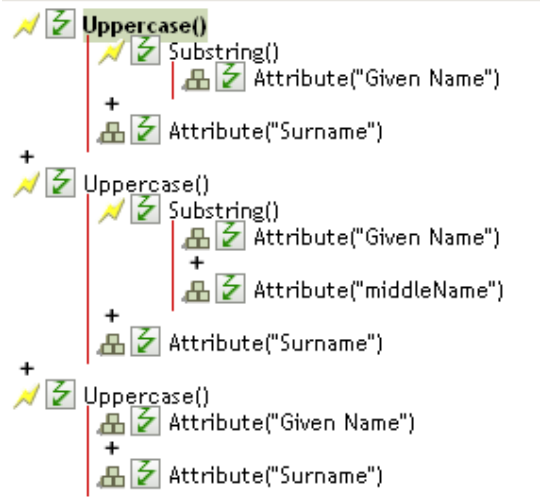
Example



The following is an example of the Editor pane when constructing the unique name argument:



The following pattern was constructed to provide unique names:



If this pattern does not generate a unique name, a digit is appended, incrementing up to the specified number of digits. In this example, nine additional unique names would be generated by the appended digit before an error occurs (pattern1 - pattern99).

Unmatched Source DN

Expands to the part of the source DN in the current operation that corresponds to the part of the DN that was not matched by the most recent match of an If Source DN condition.

Fields

Convert

Select whether or not to convert the DN format used by the destination data store.

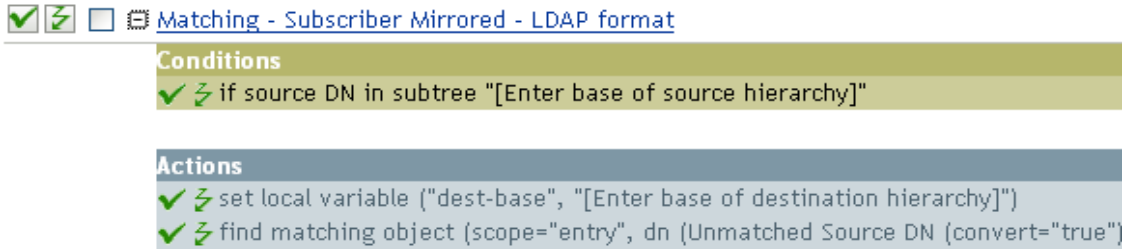
Remarks

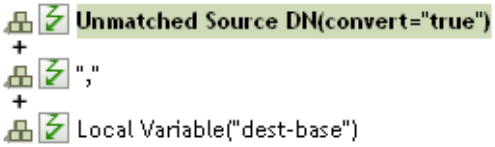
If there are no matches, the entire DN is used.

Example

The example is from the predefined rules that come with Identity Manager. For more information, see “Matching - Subscriber Mirrored - LDAP Format” on page 61. To view the policy in XML, see predef_match_sub_mirrored.xml (../samples/predef_match_sub_mirrored.xml).

The action of Finding Matching Object uses the Unmatched Source DN token to build the matching information in LDAP format. It takes the unmatched portion of the source DN to make a match.





XPath

Expands to results of evaluating an XPath 1.0 expression.

Fields

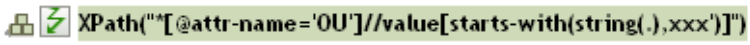
Expression

XPath 1.0 expression to evaluate.

Remarks

For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *Understanding Policies for Identity Manager 3.6*.

Example



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 from $<variable-name>$ is replaced with the value of the local or global variable with the five name. $<variable-name>$ must be a legal variable name. For more information on what is a legal XML name, see W3C Extensible Markup Language (http://www.w3.org/TR/2006/REC-xml11-20060816/#sec-suggested-names).

Verb Tokens

12

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 267).

Example

Base64 Encode

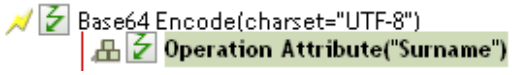
Converts the result of the enclosed tokens to bytes using the specified character set, and then Base64-encodes the bytes.

Fields

Character Set

Specify the character set that converts the string to bytes. It can be any Java supported character set. If the 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 267).

Example



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 267).

Source Language

Specify the source language (defaults to the current system language). Supports variable expansion. For more information, see Variable Expansion (page 267).

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 267).

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 267).

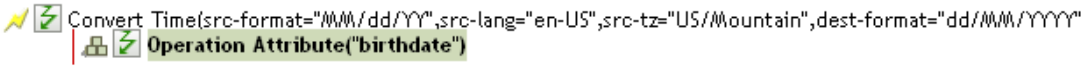
Destination Language

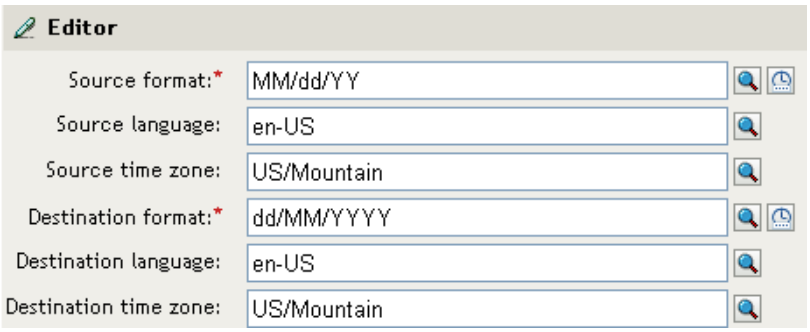
Specify the destination language (defaults to the current system language). Supports variable expansion. For more information, see Variable Expansion (page 267).

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 267).

Example





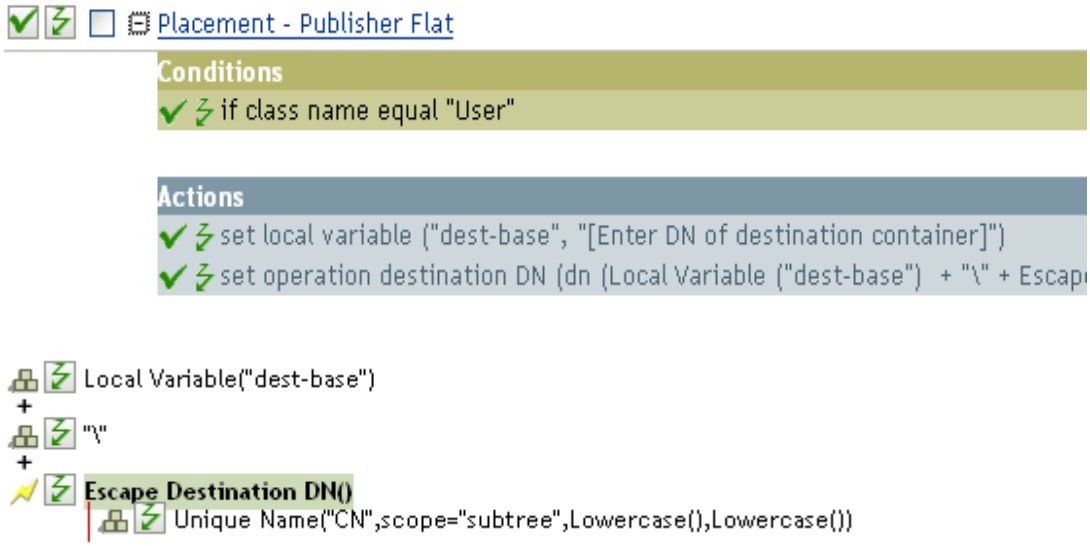
Escape Destination DN

Escapes the enclosed tokens according to the rules of the DN format of the destination data store.

Example

The example is from the predefined rules that come with Identity Manager. For more information, see Section 6.16, “Placement - Publisher Flat,” on page 66. To view the policy in XML, see predef_place_pub_flat.xml (../samples/predef_place_pub_flat.xml).

The action of Set Operation Destination DN uses the Escape Destination DN token to build the destination DN of the User object.

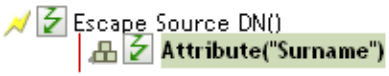


The Escape Destination DN token takes the value in Unique Name and sets it to the format for the destination DN.

Escape Source DN

Escapes the enclosed tokens according to the rules of the DN format of the source data store.

Example



Join

Joins the values of the nodes in the node set result of the enclosed tokens, separating the values by the characters specified by delimiter. If the comma-separated values (CSV) are true, then CSV quoting rules are applied to the values.

Fields

Delimiter

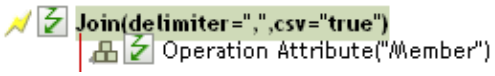
(Optional) Specify the string used to delimit the joined values. Supports variable expansion. For more information, see Variable Expansion (page 267).

Apply CSV Quoting Rules

Applies CSV quoting values.

Example

The example combines all of the members of the group into a CSV record.

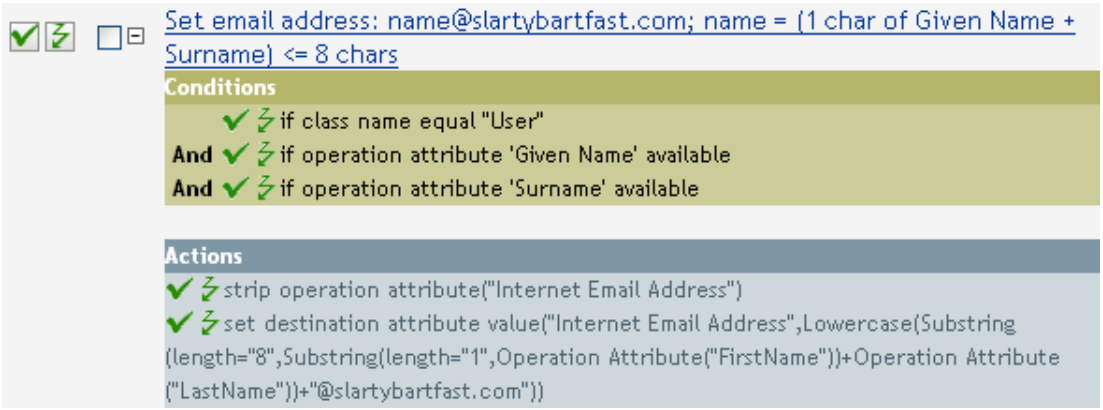


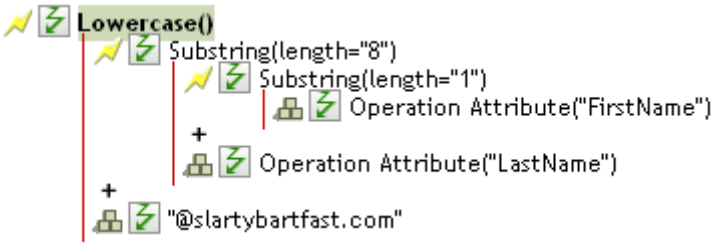
Lowercase

Converts the characters in the enclosed tokens to lowercase.

Example

This example sets the e-mail address to be name@slartybartfast.com where the name equals the first character of the Given Name plus the Surname. The policy name is Policy: Create E-mail from Given Name and Surname, and it is available for download at the Novell Support Web site. For more information, see "Downloading Identity Manager Policies" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 001-Command-SetEmailByGivenNameAndSurnam.xml (../samples/001-Command-SetEmailByGivenNameAndSurname.xml).





The Lowercase token sets all of the information in the action Set Destination attribute value to lowercase.

Map

Maps the result of the enclosed tokens from the values specified by the source column to the destination column in the specified mapping table.

Remarks

If this token is evaluated in a context where a node set result is expected and multiple rows are matched by the value being mapped, a node set is returned that contains the values from the destination column of each matching row. Otherwise, only the value from the first matching row is returned.

The table attribute should be the slash form DN of the Resource object containing the mapping table to be used. The DN might be relative to the including policy.

Fields

Mapping Table DN

Specify the slash form DN of a Resource object containing the mapping table. Supports variable expansion. For more information, see Variable Expansion (page 267).

Render Browse DN Relative to Policy

When it is enabled, it displays the mapping table DN relative to the policy. This is the default.

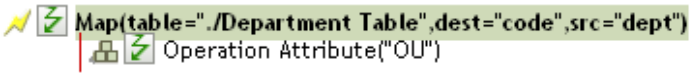
Source Column Name

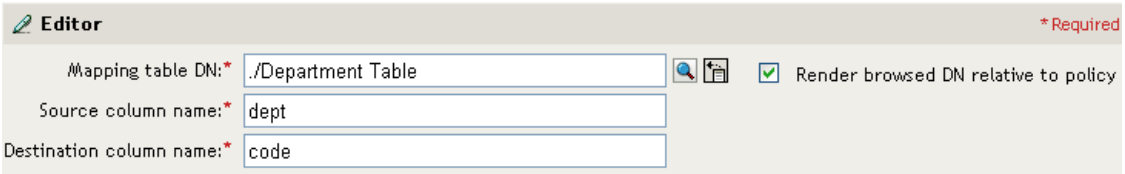
Specify the name of the source column. Supports variable expansion. For more information, see Variable Expansion (page 267).

Destination Column Name

Specify the name of the destination column. Supports variable expansion. For more information, see Variable Expansion (page 267).

Example





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 segment 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

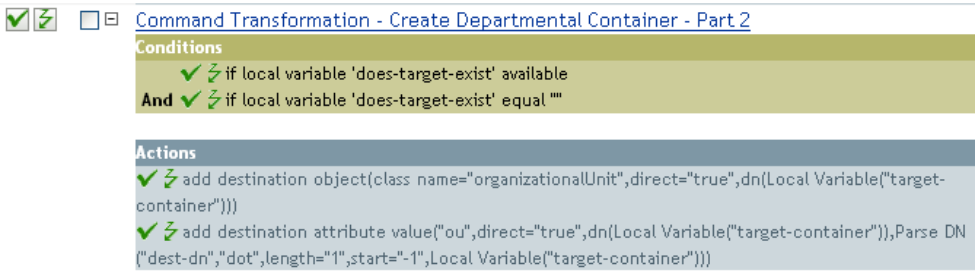
- Escape Character

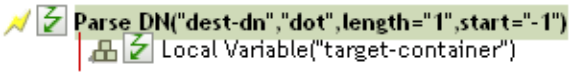
If RDN Delimiter and Relative RDN Delimiter are the same character, the orientation of the name is root right, otherwise the orientation is root left.

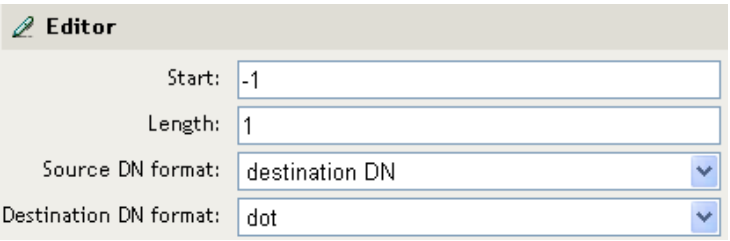
If there are more than eight characters in the delimiter set, the extra characters are considered as characters that need to be escaped, but they have no other special meaning.

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" on page 48. To view the policy in XML, see predef_command_create_dept_container2.xml (../samples/predef_command_create_dept_container2.xml).







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 267).

Replace With

Specify the replacement string. Supports variable expansion. For more information, see Variable Expansion (page 267).

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 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 267).

Replace With

Specify the replacement string. Supports variable expansion. For more information, see Variable Expansion (page 267).

Remarks

The matching instance is replaced by the string specified in the *Replace with field*.

For details on creating regular expressions, see:

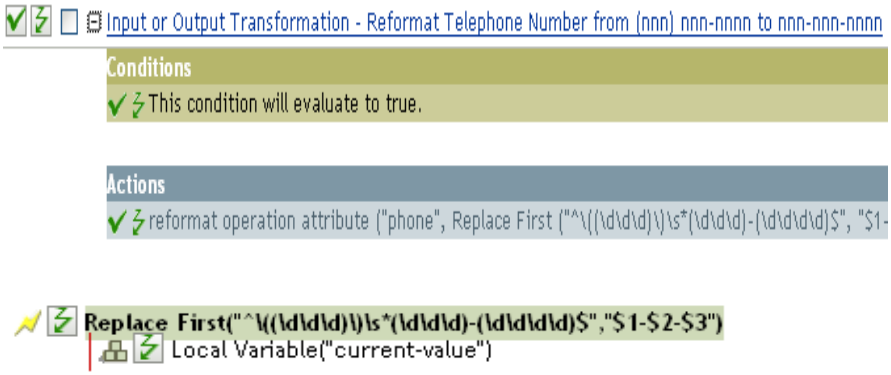
- Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html)
- Sun's Web site (java.lang.String) (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Matcher.html#replaceAll (java.lang.String))

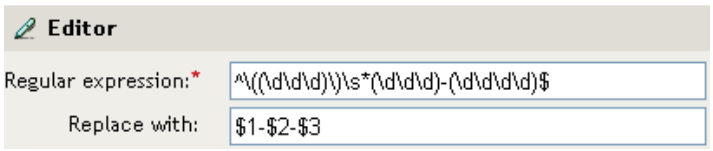
The pattern option CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.

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" on page 57. 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.





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

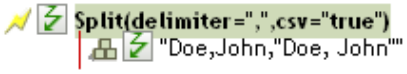
Delimiter

Regular expression that matches the delimiter characters. Supports variable expansion. For more information, see Variable Expansion (page 267).

Apply CSV Quoting Rules

Applies CSV quoting values.

Example



Substring

Extracts a portion of the enclosed tokens.

Fields

Start

Specify the starting character index:

- Index 0 is the first character.
- Positive indexes are an offset from the start of the string.
- Index -1 is the last character.
- Negative indexes are an offset from the last character toward the start of the string.

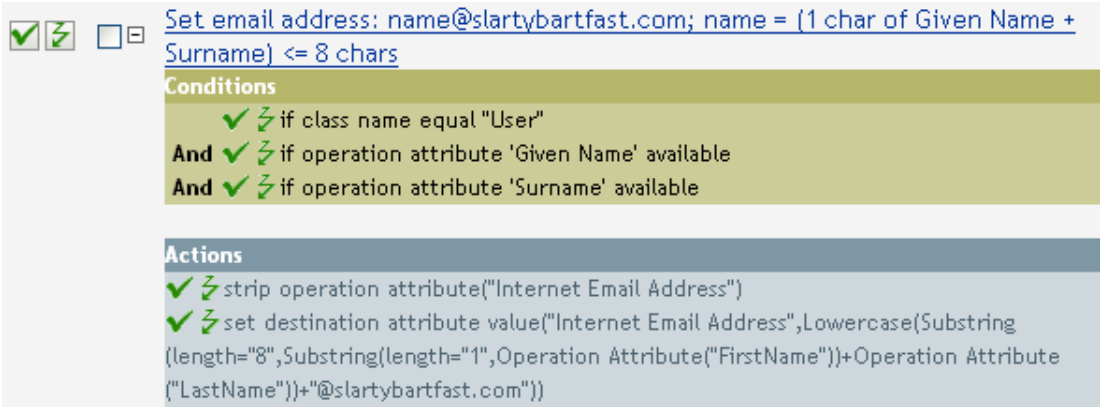
For example, if the start is specified as -2, then it starts reading the first character from the end. If -3 is specified, then is starts 2 characters from the end.

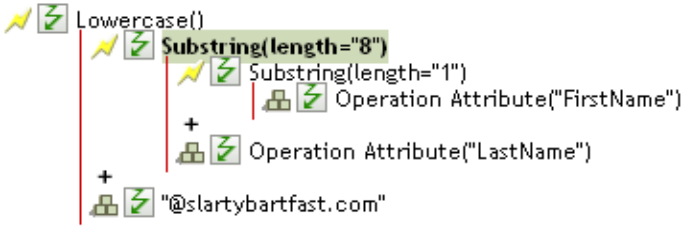
Length

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 001-Command-SetEmailByGivenNameAndSurname.xml (../samples/001-Command-SetEmailByGivenNameAndSurname.xml).





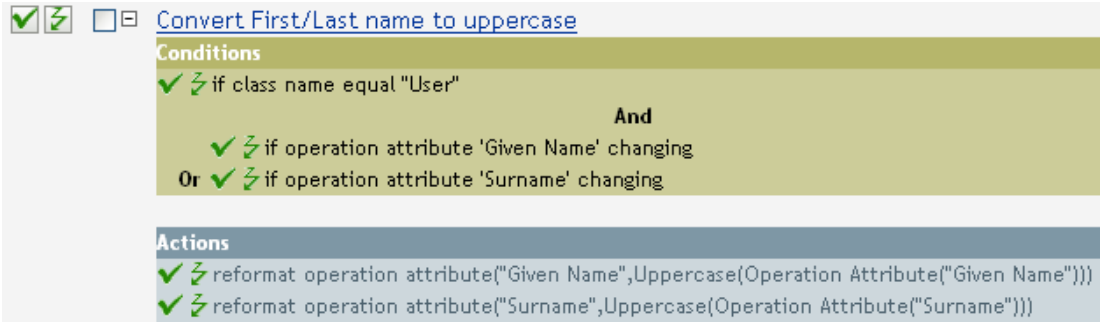
The Substring token is used twice in the action Set Destination Attribute Value. It takes the first character of the First Name attribute and adds eight characters of the Last Name attribute together to form one substring.

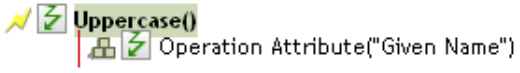
Uppercase

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" in *Understanding Policies for Identity Manager 3.6*. To view the policy in XML, see 002-Command-UppercaseNames.xml (../samples/002-Command-UppercaseNames.xml).

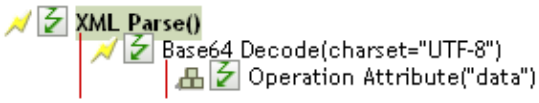




XML Parse

Parses the result of the enclosed tokens as XML and returns the resulting document node in a node set. If the result of the enclosed tokens is not well-formed XML or cannot be parsed for any reason, an empty node set is returned.

Example



XML 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

Variable Expansion

Allows for the use of dynamic variables in the verb token.

Remark

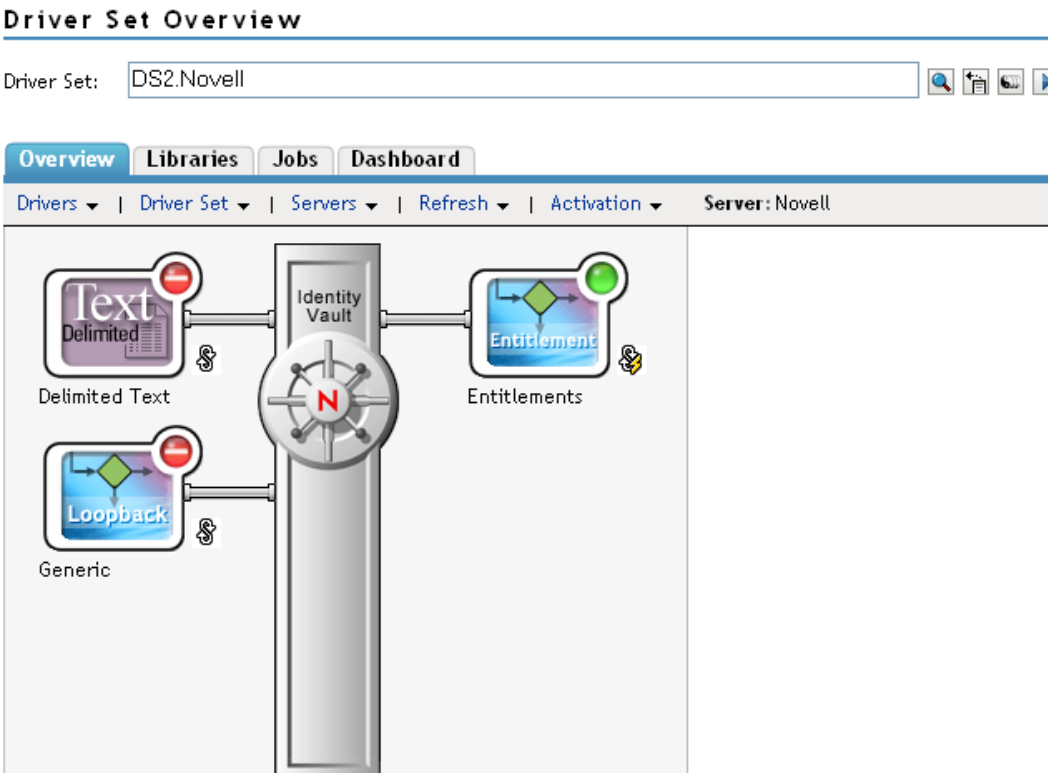
Many verb tokens support dynamic variable expansion in their attributes or content. Where supported, an embedded reference of the from $<variable-name>$ is replaced with the value of the local or global variable with the five name. $<variable-name>$ must be a legal variable name. For more information on what is a legal XML name, see W3C Extensible Markup Language (http://www.w3.org/TR/2006/REC-xml11-20060816/#sec-suggested-names).

iManager Navigation

A

A.1 Accessing the Identity Manager Driver Set Overview Page

1. In iManager, click  to display the Identity Manager Administration page.
2. In the *Administration* list, click *Identity Manager Overview* to display the Identity Manager Overview page.
3. In the *Search in* field, specify the fully distinguished name of the container where you want to start searching and then click  , or click  to browse for and select the container in the tree structure.
4. After the search completes and displays the driver sets, click the desired driver set to display the Driver Set Overview page.



A.2 Accessing the Identity Manager Driver Overview Page

1. In iManager, click  to display the Identity Manager Administration page.
2. In the *Administration* list, click *Identity Manager Overview* to display the Identity Manager Overview page.
3. In the *Search in* field, specify the fully distinguished name of the container where you want to start searching for the driver set and then click  , or click  to browse for and select the container in the tree structure.
4. After the search completes and displays the driver sets, click the driver set in which the driver resides to display the Driver Set Overview page.
5. Click the desired driver. The Identity Manager Driver Overview page opens.