Policy Guide

Access Manager 3.2

May 2012

Legal Notice

THIS DOCUMENT AND THE SOFTWARE DESCRIBED IN THIS DOCUMENT ARE FURNISHED UNDER AND ARE SUBJECT TO THE TERMS OF A LICENSE AGREEMENT OR A NON-DISCLOSURE AGREEMENT. EXCEPT AS EXPRESSLY SET FORTH IN SUCH LICENSE AGREEMENT OR NON-DISCLOSURE AGREEMENT, NETIQ CORPORATION PROVIDES THIS DOCUMENT AND THE SOFTWARE DESCRIBED IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. SOME STATES DO NOT ALLOW DISCLAIMERS OF EXPRESS OR IMPLIED WARRANTIES IN CERTAIN TRANSACTIONS; THEREFORE, THIS STATEMENT MAY NOT APPLY TO YOU.

This document and the software described in this document may not be lent, sold, or given away without the prior written permission of NetIQ Corporation, except as otherwise permitted by law. Except as expressly set forth in such license agreement or non-disclosure agreement, no part of this document or the software described in this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, or otherwise, without the prior written consent of NetIQ Corporation. Some companies, names, and data in this document are used for illustration purposes and may not represent real companies, individuals, or data.

This document could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein. These changes may be incorporated in new editions of this document. NetIQ Corporation may make improvements in or changes to the software described in this document at any time.

© 2012 NetIQ Corporation and its affiliates. All Rights Reserved.

U.S. Government Restricted Rights: If the software and documentation are being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), in accordance with 48 C.F.R. 227.7202-4 (for Department of Defense (DOD) acquisitions) and 48 C.F.R. 2.101 and 12.212 (for non-DOD acquisitions), the government's rights in the software and documentation, including its rights to use, modify, reproduce, release, perform, display or disclose the software or documentation, will be subject in all respects to the commercial license rights and restrictions provided in the license agreement.

Check Point, FireWall-1, VPN-1, Provider-1, and SiteManager-1 are trademarks or registered trademarks of Check Point Software Technologies Ltd.

Access Manager, ActiveAudit, ActiveView, Aegis, AppManager, Change Administrator, Change Guardian, Cloud Manager, Compliance Suite, the cube logo design, Directory and Resource Administrator, Directory Security Administrator, Domain Migration Administrator, Exchange Administrator, File Security Administrator, Group Policy Administrator, Group Policy Guardian, Group Policy Suite, IntelliPolicy, Knowledge Scripts, NetConnect, NetIQ, the NetIQ logo, PlateSpin, PlateSpin Recon, Privileged User Manager, PSAudit, PSDetect, PSPasswordManager, PSSecure, Secure Configuration Manager, Security Administration Suite, Security Manager, Server Consolidator, VigilEnt, and Vivinet are trademarks or registered trademarks of NetIQ Corporation or its affiliates in the USA. All other company and product names mentioned are used only for identification purposes and may be trademarks or registered trademarks of their respective companies.

For purposes of clarity, any module, adapter or other similar material ("Module") is licensed under the terms and conditions of the End User License Agreement for the applicable version of the NetIQ product or software to which it relates or interoperates with, and by accessing, copying or using a Module you agree to be bound by such terms. If you do not agree to the terms of the End User License Agreement you are not authorized to use, access or copy a Module and you must destroy all copies of the Module and contact NetIQ for further instructions.

About This Guide

This guide describes the following features of NetIQ Access Manager policies:

- Chapter 1, “Managing Policies,” on page 9
- Chapter 2, “Creating Role Policies,” on page 23
- Chapter 3, “Creating Authorization Policies,” on page 65
- Chapter 4, “Creating Identity Injection Policies,” on page 113
- Chapter 5, “Creating Form Fill Policies,” on page 131
- Chapter 6, “Creating External Attribute Source Policies,” on page 159
- Chapter 7, “Troubleshooting Access Manager Policies,” on page 161

This is intended to help you understand and configure all of the policy features provided by Access Manager and includes advanced topics.

It is recommended that you first become familiar with the information in the *NetIQ Access Manager 3.2 Setup Guide*, which helps you understand how to perform a basic Identity Server configuration, set up a resource protected by an Access Gateway, and configure SSL.

Audience

This guide is intended for Access Manager administrators. It is assumed that you have knowledge of evolving Internet protocols, such as:

- Extensible Markup Language (XML)
- Simple Object Access Protocol (SOAP)
- Security Assertion Markup Language (SAML)
- Public Key Infrastructure (PKI) digital signature concepts and Internet security
- Secure Socket Layer/Transport Layer Security (SSL/TLS)
- Hypertext Transfer Protocol (HTTP and HTTPS)
- Uniform Resource Identifiers (URIs)
- Domain Name System (DNS)
- Web Services Description Language (WSDL)

Feedback

We want to hear your comments and suggestions about this guide and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation.

Documentation Updates

For the most recent version of the *Access Manager Policies Guide*, visit the NetIQ Access Manager Documentation Web site (https://www.netiq.com/documentation/novellaccessmanager32/).

Additional Documentation

Before proceeding, you should be familiar with the *NetIQ Access Manager 3.2 Installation Guide* and the *NetIQ Access Manager 3.2 Setup Guide*, which provide information about installing and setting up the Access Manager system.

The following device reference guides are also available:

- *NetIQ Access Manager 3.2 Identity Server Guide*
- *NetIQ Access Manager 3.2 Administration Console Guide*
- *NetIQ Access Manager 3.2 Access Gateway Guide*
- *NetIQ Access Manager 3.2 SSL VPN Server Guide*
- *NetIQ Access Manager 3.2 J2EE Agent Guide*

Contents

About This Guide **3**

1 Managing Policies **9**

1.1 Selecting a Policy Type . . . 9
1.2 Policy Performance . . . 10
1.3 Managing Policies . . . 11
1.3.1 Creating Policies . . . 11
1.3.2 Sorting Policies . . . 11
1.3.3 Deleting Policies . . . 11
1.3.4 Renaming or Copying a Policy . . . 12
1.3.5 Importing and Exporting Policies . . . 12
1.3.6 Creating the SSL VPN Default Policy . . . 12
1.3.7 Refreshing Policy Assignments . . . 12
1.3.8 Viewing Policy Information . . . 12
1.4 Viewing Policies . . . 13
1.5 Managing Policy Containers . . . 13
1.6 Managing a Rule List . . . 14
1.6.1 Rule Evaluation for Role Policies . . . 15
1.6.2 Rule Evaluation for Authorization Policies . . . 15
1.6.3 Rule Evaluation for Identity Injection and Form Fill Policies . . . 15
1.6.4 Viewing Rules . . . 16
1.7 Adding Policy Extensions . . . 16
1.7.1 Installing the Extension on the Administration Console . . . 17
1.7.2 Distributing a Policy Extension . . . 19
1.7.3 Managing a Policy Extension Configuration . . . 20
1.7.4 Viewing Extension Details . . . 20
1.8 Enabling Policy Logging . . . 20

2 Creating Role Policies **23**

2.1 Understanding RBAC in Access Manager . . . 23
2.1.1 Assigning All Authenticated Users to a Role . . . 24
2.1.2 Using a Role to Create an Authentication Policy . . . 25
2.1.3 Using Prioritized Rules in an Authorization Policy . . . 27
2.2 Creating Roles . . . 27
2.2.1 Selecting Conditions . . . 29
2.2.2 Using Multiple Conditions . . . 42
2.2.3 Selecting an Action . . . 44
2.3 Example Role Policies . . . 46
2.3.1 Creating an Employee Role . . . 47
2.3.2 Creating a Manager Role . . . 48
2.3.3 Creating a Rule for a Contract with ORed Credentials . . . 50
2.4 Creating Access Manager Roles in an Existing Role-Based Policy System . . . 52
2.4.1 Activating Roles from External Sources . . . 52
2.4.2 Using Conditions to Assign Roles . . . 54
2.5 Mapping Roles between Trusted Providers . . . 61
2.5.1 Prerequisites . . . 61
2.5.2 Procedure . . . 62
2.6 Enabling and Disabling Role Policies . . . 63
2.7 Importing and Exporting Role Policies . . . 63

3 Creating Authorization Policies 65

3.1 Designing an Authorization Policy . . . 65
3.1.1 Controlling Access with a Deny Rule and a Negative Condition . . . 66
3.1.2 Configuring the Result on Condition Error Option . . . 67
3.1.3 Many Rules or Many Conditions . . . 67
3.1.4 Using Multiple Conditions . . . 67
3.1.5 Controlling Access with Multiple Conditions . . . 69
3.1.6 Using Permit Rules with a Deny Rule . . . 70
3.1.7 Using Deny Rules with a General Permit Rule . . . 71
3.1.8 Public Policies . . . 73
3.1.9 General Design Principles . . . 73
3.1.10 Using the Refresh Data Option . . . 74
3.1.11 Assigning Policies to Resources . . . 74
3.2 Creating Access Gateway Authorization Policies . . . 75
3.3 Sample Access Gateway Authorization Policies . . . 77
3.3.1 Sample Policy Based on Organizational Rules . . . 77
3.3.2 Sample Workflow Policy . . . 81
3.4 Creating Web Authorization Policies for J2EE Agents . . . 84
3.5 Creating Enterprise JavaBean Authorization Policies for J2EE Agents . . . 86
3.6 Conditions . . . 87
3.6.1 Authentication Contract Condition . . . 88
3.6.2 Client IP Condition . . . 90
3.6.3 Credential Profile Condition . . . 91
3.6.4 Current Date Condition . . . 92
3.6.5 Day of Week Condition . . . 93
3.6.6 Current Day of Month Condition . . . 95
3.6.7 Current Time of Day Condition . . . 95
3.6.8 HTTP Request Method Condition . . . 97
3.6.9 LDAP Attribute Condition . . . 98
3.6.10 LDAP OU Condition . . . 99
3.6.11 Liberty User Profile Condition . . . 100
3.6.12 Roles Condition . . . 101
3.6.13 URL Condition . . . 102
3.6.14 URL Scheme Condition . . . 103
3.6.15 URL Host Condition . . . 105
3.6.16 URL Path Condition . . . 106
3.6.17 URL File Name Condition . . . 107
3.6.18 URL File Extension Condition . . . 108
3.6.19 X-Forward-For IP Condition . . . 110
3.6.20 Condition Extension . . . 111
3.6.21 Data Extension . . . 111
3.6.22 Using the URL Dredge Option . . . 111
3.6.23 Edit Button . . . 112
3.7 Importing and Exporting Authorization Policies . . . 112

4 Creating Identity Injection Policies 113

4.1 Designing an Identity Injection Policy . . . 113
4.1.1 Using the Refresh Data Option . . . 114
4.2 Configuring an Identity Injection Policy . . . 115
4.3 Configuring an Authentication Header Policy . . . 116
4.4 Configuring a Custom Header Policy . . . 120
4.5 Configuring a Custom Header with Tags . . . 123
4.6 Specifying a Query String for Injection . . . 125
4.7 Injecting into the Cookie Header . . . 128
4.8 Importing and Exporting Identity Injection Policies . . . 128
4.9 Sample Identity Injection Policy . . . 129

5 Creating Form Fill Policies 131

5.1 Understanding an HTML Form . . . 131
5.2 Creating a Form Fill Policy for the Sample Form . . . 134
5.3 Implementing Form Fill Policies . . . 137
5.3.1 Designing a Form Fill Policy . . . 137
5.3.2 Creating a Form Fill Policy . . . 142
5.3.3 Creating a Login Failure Policy . . . 147
5.3.4 Troubleshooting a Form Fill Policy . . . 148
5.4 Creating and Managing Shared Secrets . . . 150
5.4.1 Naming Conventions for Shared Secrets . . . 151
5.4.2 Creating a Shared Secret Independent of a Policy . . . 151
5.4.3 Modifying and Deleting a Shared Secret . . . 152
5.5 Importing and Exporting Form Fill Policies . . . 152
5.6 Configuring a Form Fill Policy for Forms With Scripts . . . 153
5.6.1 Why Does Form Fill Fail with the Default Policy? . . . 153
5.6.2 Understanding How a Form Is Submitted . . . 155
5.6.3 Creating a Form Fill Policy for Autosubmission . . . 156
5.6.4 Configuring the Advanced Options for Autosubmission . . . 157

6 Creating External Attribute Source Policies 159

6.1 Configuring an External Attribute Source Policy . . . 159

7 Troubleshooting Access Manager Policies 161

7.1 Turning on Logging for Policy Evaluation . . . 161
7.2 Understanding Policy Evaluation Traces . . . 162
7.2.1 Format . . . 163
7.2.2 Policy Result Values . . . 169
7.2.3 Role Assignment Traces . . . 170
7.2.4 Identity Injection Traces . . . 171
7.2.5 Authorization Traces . . . 173
7.2.6 Form Fill Traces . . . 175
7.3 Common Configuration Problems That Prevent a Policy from Being Applied as Expected . . . 180
7.3.1 Enabling Roles for Authorization Policies . . . 180
7.3.2 LDAP Attribute Condition . . . 181
7.3.3 Result on Condition Error Value . . . 181
7.3.4 An External Secret Store and Form Fill . . . 182
7.4 The Policy Is Using Old User Data . . . 182
7.5 Form Fill and Identity Injection Silently Fail . . . 183
7.6 Checking for Corrupted Policies . . . 184
7.7 Policy Page Timeout . . . 184
7.8 Policy Creation and Storage . . . 184
7.9 Policy Distribution . . . 184
7.10 Policy Evaluation: Access Gateway Devices . . . 185
7.10.1 Successful Policy Configuration Example . . . 186
7.10.2 No Policy Defined Configuration Example . . . 186
7.10.3 Deny Access Configuration/Evaluation Example . . . 187

1 Managing Policies

Policies are logical and testable rules that you use to maintain order, security, and consistency within your Access Manager infrastructure. You can specify activation criteria, deactivation criteria, temporal constraints (such as time of day or subnet), identity constraints (such as user object attribute values), and additional separation-of-duty constraints. Identity information can come from any identity source (such LDAP, an Identity Vault, or a directory) or from the Access Manager's Identity Server, which provides full Liberty Alliance specification support and SAML 2.0 support. Identity is available throughout the determination of rights and permissions.

- Section 1.1, "Selecting a Policy Type," on page 9
- Section 1.2, "Policy Performance," on page 10
- Section 1.3, "Managing Policies," on page 11
- Section 1.4, "Viewing Policies," on page 13
- Section 1.5, "Managing Policy Containers," on page 13
- Section 1.6, "Managing a Rule List," on page 14
- Section 1.7, "Adding Policy Extensions," on page 16
- Section 1.8, "Enabling Policy Logging," on page 20

1.1 Selecting a Policy Type

Access Manager uses the policy type to define the context within which a policy is evaluated. Each type of policy differs in purpose, which in turn determines the conditions and actions that apply. For example, the conditions and actions of an Authorization policy differ from the conditions and actions of an Identity Injection policy.

When you click *New* on the Policies page, the system displays the predefined policy types in a drop-down list. Each policy type represents the set of conditions and actions that are available. You then configure rules to determine user roles, make decision requests, and enforce authorization decisions. You can also set up policies with no conditions, allowing actions to always take place. As policies and conditions become complex, it can be simpler and more manageable to design policies with conditions that deny or restrict access to large groups of users, rather than setting up policies that permit access to certain users.

Access Manager has the following policy types:

- **Access Gateway: Authorization:** This policy type is used to permit or deny access to protected resources, such as Web servers. After you have set up the protected resource, you use the policy rules to define how you want to restrict access. For example, if a user is denied access to a resource, you can use the policy to redirect them to a URL where they can request access to the resource.

- **Access Gateway: Identity Injection:** This policy type evaluates the rules for Identity Injection, which retrieves identity data from a data source (user store) and forwards it to Web applications. Such a policy can enable single sign-on. After the user has authenticated, the policy supplies the information required by the resource rather than allowing the resource to prompt the user for the information.
- **Access Gateway: Form Fill:** This policy type is used to create a policy that automatically fills in the information required in a form, after the user has filled in the form once. Such a policy can enable single sign-on to resources that require form data before allowing access.
- **Identity Server: Roles:** This policy type evaluates rules for establishing the roles of an authenticated user. Roles are generated based on policy statements each time a user authenticates. Roles are placed into an Authentication Profile, which can be used as input in policies for Authorization or Identity Injection.
- **Identity Server: External Attribute Source:** This policy type is used to create a policy that retrieves the attributes from external sources.
- **J2EE Agent: EJB Authorization:** This policy type allows you to create policies that protect an Enterprise JavaBean. You can protect the entire bean or specific interfaces or methods.
- **J2EE Agent: Web Authorization:** This policy type allows you to create policies that protect the Web applications on a J2EE server.

1.2 Policy Performance

Authorization and Identity Injection policies allow you to select conditions, one of which is Roles. If you have thousands of users accessing your resources, you might want to design most of your policies to use roles. Roles are evaluated when a user logs in, and the roles assigned to the user are cached as long as the session is active. When the user accesses a resource protected by a policy that uses role conditions, the policy can be immediately evaluated because the user's role values are available. This is not true for all conditions; the values for some conditions must be retrieved from the user store. For example, if the policy uses a condition with an LDAP attribute, the user's value must be retrieved from the LDAP user store before the policy can be evaluated. On a system with medium traffic, this delay won't be noticed. On a system with high traffic, the delay might be noticeable.

However, you can design your policies to have the same results without causing the retrieval of the LDAP attribute value at resource access. You can create a Role policy for the LDAP attribute and have users assigned to this role at authentication when they match the attribute value requirements. When the users access the resources, they gain immediate access (or are immediately denied access) because their role assignments are cached.

If the same LDAP attribute policy is used to grant access to multiple resources, the chance that the user notices a delay is slight. The first time a policy is evaluated for a user, the data required for the policy is cached and is therefore immediately available the next time it is requested.

Another option available for LDAP, Credential Profile, Liberty User Profile, and Shared Secret attributes is to have the attribute values sent with the assertion at authentication. You configure an attribute set for the attributes, and then configure the service provider for these attributes. For more information, see "Configuring the Attributes Sent with Authentication" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

As you design your policies, experiment and find the type that works best for your network and your customers.

1.3 Managing Policies

1. In the Administration Console, click *Policies* > *Policies*.
2. In the Policy Container drop-down list, select the container.

   If you have not created any containers, only the Master_Container is available in the list.
3. You can perform the following tasks from this page:
   - "Creating Policies" on page 11
   - "Sorting Policies" on page 11
   - "Deleting Policies" on page 11
   - "Renaming or Copying a Policy" on page 12
   - "Importing and Exporting Policies" on page 12
   - "Creating the SSL VPN Default Policy" on page 12
   - "Refreshing Policy Assignments" on page 12

1.3.1 Creating Policies

Before creating policies, you need to design your policy strategy. For example, if you are going to use role-based access, you need to decide which roles you need and which roles allow access to your protected resources. Roles, which are used by Authorization policies that grant and deny access, need to be created first. If you have already created the roles and assigned them to users in your LDAP user store, you can use the values of your role attributes in the Authorization policies rather than using Access Manager roles.

To create a policy, see the following sections:

- Chapter 2, "Creating Role Policies," on page 23
- Chapter 3, "Creating Authorization Policies," on page 65
- Chapter 4, "Creating Identity Injection Policies," on page 113
- Chapter 5, "Creating Form Fill Policies," on page 131
- Chapter 6, "Creating External Attribute Source Policies," on page 159

1.3.2 Sorting Policies

Policies can be sorted by name and by type. On the Policies page, click *Name* in the *Policy List*, and the policies are sorted alphabetically by name. To sort alphabetically by type, click *Type* in the *Policy List*.

You can also use containers to organize your policies. For more information, see Section 1.5, "Managing Policy Containers," on page 13.

1.3.3 Deleting Policies

A policy cannot be deleted as long as a resource is configured to use the policy. This means that you must remove the policy from all protected resources for the Access Gateway and J2EE Agent policies.

Roles can be used by Authorization, Form Fill, and Identity Injection policies. Before you can delete a Role policy, you must remove any reference to the role from all other policies.

1.3.4 Renaming or Copying a Policy

Copy: To copy a policy, select a policy, click *Copy*, then click *OK*. The new policy is named "Copy of ..." This is useful when you are creating multiple policies that require only minor variations to make them unique. You should rename the policy after making these modifications.

Rename: To rename a policy, select a policy, click *Rename*, specify a new name, then click *OK*.

1.3.5 Importing and Exporting Policies

Policies that are created in the Administration Console can be exported and used in another Administration Console that is managing a different group of Access Gateways and other devices. Each policy type has slightly different import requirements. See the following:

- Section 2.7, "Importing and Exporting Role Policies," on page 63
- Section 3.7, "Importing and Exporting Authorization Policies," on page 112
- Section 4.8, "Importing and Exporting Identity Injection Policies," on page 128
- Section 5.5, "Importing and Exporting Form Fill Policies," on page 152

1.3.6 Creating the SSL VPN Default Policy

To create the default policy that the SSL VPN server uses, click *Create SSL VPN Default*. This option creates an Identity Injection policy that is used to set up single sign-on with the SSL VPN server. After you have created this policy, this option is no longer available.

1.3.7 Refreshing Policy Assignments

If you have made changes in policy assignments that are not reflected on the page, click *Refresh References*. This action can take a while to complete if you have numerous policies and have assigned them to protect numerous resources. The Administration Console needs to verify the configuration of each device.

If you have made changes in policy assignments that are not reflected on the page, click *Refresh References*. This action can take a while to complete if you have numerous policies and have assigned them to protect numerous resources. The Administration Console needs to verify the configuration of each device.

1.3.8 Viewing Policy Information

The *Policy List* table displays the following information about each policy:

Column	Description
Name	Displays the name of the policy. To modify a policy, click its name.
Type	Specifies the type of policy (Authorization, Identity Injection, Roles, or Form Fill) and the type of resource that can use it (Identity Server, Access Gateway, or J2EE Agent).

Column	Description
Used By	Displays the name of the Access Gateway, the Identity Server configuration, or the J2EE Agent that the policy is assigned to. If the policy is unassigned, this column has no value. If the policy is assigned to a protected resource, click the down-arrow button to view the names of the resources it has been assigned to.
Extensions Used	Specifies whether the policy uses any extensions. If none has been used, this column has no value.
Description	Displays a description of the policy. If no description has been specified, this column has no value.

1.4 Viewing Policies

The policy view administrators can log in to Access Manager with their credentials and they are allowed to view only the policy containers assigned to them.

1. Log in to Access Manager.
2. In the Policy Container drop-down list, select the container.

The *Policy List* table displays the following information about each policy.

Column	Description
Name	Displays the name of the policy.
Type	Specifies the type of policy (Authorization, Identity Injection, Roles, or Form Fill) and the type of resource that can use it (Identity Server, Access Gateway, or J2EE Agent).
Description	Displays a description of the policy. If no description has been specified, this column has no value.

1.5 Managing Policy Containers

You use policy containers to store and organize policies, similar to how you organize files in folders. The *Master_Container* is a permanent policy container, but you can use the *Containers* tab to create new containers.

A policy container can hold up to 500 policies. When you reach that limit, you must create another container to add, copy, or import policies. For performance and for ease in finding a policy, you might want to limit a container to 200 or fewer policies. Policies in a container can be sorted by name and type, to aid you in finding a particular policy.

If you have only one administrator configuring and managing policies, you can create additional policy containers to help you keep policies organized. If you have multiple administrators creating policies, you can create a container for each administrator to use. This allows multiple administrators to modify policies at the same time. When an administrator opens a policy in a container, the container is locked, which prevents other administrators from modifying any policies in that container until changes are applied or canceled.

1. In the Administration Console, click *Policies* >*Containers*.
2. On the Containers page, click *New*.

3 Name the policy container, then click *OK*.

4 Click *Close*.

After you add a policy container, the system displays it in the *Policy Container* drop-down list on the Policy List page.

You must delete all the policies in a policy container before you can delete the policy container.

1 Select the check box of the policy container. Click *Delete*.A Confirm dialog box displays a message "Number of Containers selected: 1, Delete selected Containers?"

2 Click *Ok* to continue or *Cancel* to close the window

1.6 Managing a Rule List

You configure rules to create a policy. The rules collectively represent a desired course of action when the required conditions are met, such as denying entry-level employees access to a secure Web site, and permitting access for employees who have a role of Manager.

When the system evaluates the policy conditions, it begins with the rule with the highest priority and evaluates the conditions, starting with the first condition group in the rule. Each rule contains one or more conditions and one or more actions. If a rule's conditions are met, the rule's action is performed. For some policy types, the performance of any rule's action terminates the policy evaluation. With Authorization policies, for example, after the policy has determined that a user is either permitted or denied access to a resource, there is no reason to evaluate the policy further. However, a Role policy might identify multiple roles to which a user belongs. In this case, each rule of the policy must be evaluated to determine all roles to which the user belongs.

IMPORTANT: The interface for the policy engine is designed for flexibility. It does not protect you from creating rules that do nothing because they are always true or always false. For example, you can set up a condition where Client IP is equal to Client IP, which is always true. You are responsible for defining the condition so that it does a meaningful comparison.

To manage the list of rules for a policy:

1 In the Administration Console, click *Policies* > *Policies*.

2 Select the container.

3 Click the name of the policy.

4 In the *Rule List* section, select one of the following:

New: To create a new rule, click *New*.

You use multiple rules to coordinate how a policy operates, and the behavior varies according to the policy type. To understand how multiple rules are evaluated, see the following:

- Section 1.6.1, "Rule Evaluation for Role Policies," on page 15
- Section 1.6.2, "Rule Evaluation for Authorization Policies," on page 15
- Section 1.6.3, "Rule Evaluation for Identity Injection and Form Fill Policies," on page 15

Delete: Select a rule, then click this option to delete the rule. If the policy has only one rule, you cannot delete the last rule.

Copy: Select a rule, then click this option to copy a rule. To modify the copy, click the rule number.

Enable: Select a rule, then click this option to enable a rule.

Disable: Select a rule, then click this option to disable a rule.

5 Click *OK*, then click *Apply Changes*.

1.6.1 Rule Evaluation for Role Policies

A Role policy is used to determine which role or roles a user is assigned to. However, you can specify only one role per rule. Role policies are evaluated when a user authenticates. Role policies do not directly deny or allow access to any resource, nor do they determine if a user is authenticated. A user's role can be used in the evaluation of an Authorization policy, but at that point the evaluation of the role policy has already occurred and is not directly part of the authorization process. The performance of an action (assigning a user to a role) does not terminate the evaluation of the policy, so subsequent rules in the policy continue to be evaluated.

1.6.2 Rule Evaluation for Authorization Policies

When the Access Gateway discovers a rule in an Authorization policy that either permits or denies a user access to a protected resource, it stops processing the rules in the policy. Use the following guidelines in determining whether your Authorization policy needs multiple rules:

- If the policy enforces multiple access requirements that can result in differing actions (either permit or deny), use separate rules to define the conditions and actions.
- If you want other conditions or actions processed when a rule fails, you must create a second rule for the users that fail to match the conditions.

If you create multiple rules, you can modify the order that the rules are processed. This allows you to create policies that contain a number of Permit rules that allow access if the user matches the rule. The lowest priority rule in such a policy is a Deny rule, which denies access to everyone who has not previously matched a Permit rule.

IMPORTANT: If you create policies with multiple Permit rules, you should make the last rule in the policy a generic deny policy (a rule with no conditions and with an action of deny). This ensures that if the Result on Error Condition field in a rule is set incorrectly, the user matches the last rule and is denied access. Without this rule, a user might gain access because the user didn't match any of the rules.

You can also create a number of policies and enable multiple policies for the same protected resource. Rule priority determines how the enabled policies interact with each other. The rules in the policies are gathered into one list, then sorted by priority. The processing rules are applied as if the rules came from one policy. It is a personal design issue whether you create a policy with multiple rules or create multiple policies that you enable on a single protected resource. Either design produces a list of rules, sorted by priority, that is applied to the user requesting access to the protected resource.

1.6.3 Rule Evaluation for Identity Injection and Form Fill Policies

Rules in Identity Injection and Form Fill policies have actions, but no conditions. Because they have no conditions, all the rules are evaluated and the actions are performed. Identity Injection policies have two exceptions to this rule; they can insert only one authentication header and one cookie header. If you create multiple rules, each with an authentication header and a cookie header, the rule with the highest priority is processed and its actions performed. The actions in the second rule for injecting an authentication header and a cookie header are ignored.

You cannot create multiple rules for a Form Fill policy.

1.6.4 Viewing Rules

The policy view administrators can view the information related to rules.The rules collectively represent a desired course of action when the required conditions are met, such as denying entry-level employees access to a secure Web site, and permitting access for employees who have a role of Manager.

When the system evaluates the policy conditions, it begins with the rule with the highest priority and evaluates the conditions, starting with the first condition group in the rule. Each rule contains one or more conditions and one or more actions. If a rule's conditions are met, the rule's action is performed. For some policy types, the performance of any rule's action terminates the policy evaluation.

To view the list of rules for a policy:

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the container.
3. Click the name of the policy.
4. In the *Rule List* section, the policy view administrator can view the following details.:

   Type: Displays the type of the policy.

   Description: Displays the description of the policy type.

   Priority: Displays the priority of the rule number to the policy type.

   Actions: Displays the actions for the policy type.

1.7 Adding Policy Extensions

If Access Manager does not supply the action, the data type, or the condition that you need for a policy, you can add a customized policy extension. For example, suppose you need a policy that permits access based on whether a user has a specific role which is assigned to users in an Oracle database. The custom extension could read the role assignments of the user from the Oracle database and return a string containing the role names. This data could then be used to determine access rights to Access Manager resources.

For information on how to create a policy extension, see the *Novell Access Manager Developer Kit* (http://developer.novell.com/wiki/index.php/Novell_Access_Manager_Developer_Tools_and_Examples).

After a policy extension has been created, you need to perform the following tasks to use the extension:

- Section 1.7.1, "Installing the Extension on the Administration Console," on page 17
- Section 1.7.2, "Distributing a Policy Extension," on page 19

After you have configured the extension, you can perform the following tasks:

- Section 1.7.3, "Managing a Policy Extension Configuration," on page 20
- Section 1.7.4, "Viewing Extension Details," on page 20

1.7.1 Installing the Extension on the Administration Console

The policy extension can be delivered as either a `.jar` file or a `.zip` file.

- "Uploading and Configuring a JAR File" on page 17
- "Importing a ZIP File" on page 18

Uploading and Configuring a JAR File

To install an extension, you need to have access to the `.jar` file and know the following information about the extension or extensions contained within the file:

What you need to create	◆ A display name for the extension. ◆ A description for the extension.
What you need to know	◆ The policy type of the extension, which defines the policy type it can be used with. You should know whether it is an extension for an Access Gateway Authorization policy, an Access Gateway Identity Injection policy, or an Identity Server Role policy. ◆ The name of the Java class that is used by the extension. Each data type usually uses a different Java factory class. ◆ The filename of the extension. ◆ The names, IDs, and mapping type of any configuration parameters. Configuration parameters allow the policy engine to pass data to the extension, which the extension can then use to retrieve data or to evaluate a condition. ◆ The type of data the extension manipulates. **Authorization Policy:** Can be used to return the following: ◆ An action of deny, permit, or obligation. ◆ A condition that the extension evaluates and returns either true or false. ◆ A data element that the extension retrieves and the policy can use for evaluating a condition. **Identity Injection Policy:** A data extension that retrieves data for injecting into a header. **Identity Role Policy:** Can be used to return the following: ◆ A condition that the extension evaluates and returns either true or false. ◆ A data element that the extension retrieves which can be used in evaluating a condition or used to assign roles. **External Attribute Source Policy:** A data extension that retrieves attributes from external sources.

If the file contains more than one extension, you need to create a configuration for each extension in the file.

1. Copy the `.jar` file to a location that you can browse to from the Administration Console.
2. In the Administration Console, click *Policies* > *Extensions*.
3. To upload the file, click *Upload* > *Browse*, select the file, then click *Open*.

4 (Conditional) If you want this `.jar` file to overwrite an existing version of the file, select *Overwrite existing *.jar file*.

5 Click *OK*.

The file is uploaded to the Administration Console, but nothing is visible on the Extensions page until you create a configuration.

6 To create an extension configuration, click *New*, then fill in the following fields:

Name: Specify a display name for the extension.

Description: (Optional) Specify the purpose of the extension and how it should be used.

Policy Type: From the drop-down list, select the type of extension you have uploaded.

Type: From the drop-down list, select the data type of the extension.

Class Name: Specify the name of the class that creates the extension, such as com.acme.policy.action.successActionFactory.

File Name: From the drop-down list, select the `.jar` file that contains the Java class that implements the extension and its corresponding factory. This should be the file you uploaded in Step 3.

7 Click *OK*.

8 (Conditional) If the extension requires data from Access Manager, click the name of the extension.

9 In the *Configuration Parameters* section, click *New*, specify a name and ID, then click *OK*.

The developer of the extension must supply the name and ID that the extension requires.

10 In the *Mapping* column, click the down-arrow, then select the required data type.

The developer of the extension must supply the data type that is required. If the data type is a data string, then the developer needs to explain the type of information you need to supply in the text field.

11 (Conditional) If the extension requires more than one data item, repeat Step 9 and Step 10.

12 Click *OK*.

The extension is now available for the policy type it was created for.

13 (Conditional) If the class can be used for multiple policy types, you need to create an extension configuration for each policy type.

For example, if an extension can be used for both an Identity Injection policy and a Role policy, you need to create an entry for both. The *File Name* option should contain the same value, but the other options should contain unique values.

14 Continue with Section 1.7.2, "Distributing a Policy Extension," on page 19.

Importing a ZIP File

A `.zip` file with an exported extension contains both the `.jar` file and the extension configuration.

1 Copy the `.zip` file to a location that you can browse to from the Administration Console.

2 In the Administration Console, click *Policies* > *Extensions*.

3 To upload the file, click *Upload* > *Browse*, select the file, then click *Open*.

4 (Conditional) If you want the `.jar` file in the import to overwrite an existing version of the file, select *Overwrite existing *.jar file*.

5 Click *OK*.

The extension is imported in the Administration Console.

6 (Conditional) If the extension requires some customizing, click the name of the extension and follow the instructions that came with the extension.

7 Continue with Section 1.7.2, "Distributing a Policy Extension," on page 19.

1.7.2 Distributing a Policy Extension

To distributed the policy extension to the devices that need it:

1 Create a policy that uses the extension:

- **Role Policy:** To create a Role policy that uses the extension, see Section 2.2, "Creating Roles," on page 27.
- **Identity Injection Policy:** To create an Identity Injection policy that uses the extension, see Section 4.2, "Configuring an Identity Injection Policy," on page 115.
- **Authorization Policy:** To create an Authorization policy that uses the extension, see Section 3.2, "Creating Access Gateway Authorization Policies," on page 75.
- **External Attribute Source Policy:** To create an External Attribute Source policy that uses the extension, see Chapter 6, "Creating External Attribute Source Policies," on page 159.

2 Assign the policy to a device:

- For a Role policy, enable it for an Identity Server.

  For more information, see Section 2.6, "Enabling and Disabling Role Policies," on page 63.
- For an Authorization policy, assign it to a protected resource.

  For more information, see "Assigning an Authorization Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.
- For an Identity Injection policy, assign it to a protected resource.

  For more information, see "Assigning an Identity Injection Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.
- For an External Attribute Source policy, enable it for an Identity Server.

  For more information, see "Enabling External Attributes Policy" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

IMPORTANT: Do not update the device at this time. The `.jar` files must be distributed before you update the device.

---

3 Distribute the `.jar` files:

3a Click *Policies* > *Extensions*.

3b Select the extension, then click *Distribute JARs*.

3c Restart Tomcat on the devices listed for reboot.

- **Linux:** Enter the following commands:

  In the Access Gateways: `/etc/init.d/novell-mag restart`.

  In the Identity Servers: `/etc/init.d/novell-idp restart`.
- **Windows:** Enter the following commands:

```
net stop Tomcat7
net start Tomcat7
```

4 (Conditional) If the extension is for an Authorization policy or an Identity Injection policy, update the Access Gateway.

1.7.3 Managing a Policy Extension Configuration

1. In the Administration Console, click *Policies* > *Extensions*.
2. To export a policy extension, select the policy, then click *Export*.
3. To delete an extension, a policy cannot be using it. Use the *Used By* column to determine the policies that are using the extension. Modify the listed policies. When the extension is no longer used by any policies, select the extension, then click *Delete*.
4. To rename a policy extension, select the extension, click *Rename*, specify a new name, then click *OK*. When a policy extension is renamed and the extension is in use by a policy, the policy is updated. This causes the *Apply Changes* button to be active on the *Policy List* page.

1.7.4 Viewing Extension Details

You can modify the details of an existing extension and control the information Access Manager provides to the extension when the data is evaluated.

1. In the Administration Console, click *Policies* > *Extensions*.
2. Click the name of the extension.

   You can view or modify the following details:

   Description: (Optional) Specifies the purpose of the extension and how it should be used.

   Class Name: Specifies the name of the class that creates the extension, for example com.acme.policy.action.successActionFactory.

   File Name: Specifies the `.jar` file that contains the Java class that implements the extension and its corresponding factory. Select the appropriate file from the drop-down list.
3. (Conditional) Specify the Condition Parameters required by the extension.

   The documentation for the extension should tell you the number of parameters it requires and the data type of each parameter. You create the name and ID for the parameter, and they need to be unique for the extension.

   - To add a configuration parameter, click *New*, enter a name (a string) and an ID (a number) for the parameter, then click *OK*. In the *Mapping* field, click the down-arrow, then select the data item from the list. The selected data is available whenever the extension class is called to evaluate an action, a condition, or data.
   - To delete a configuration parameter, select the parameter, then click *Delete*.
4. Click *OK*.

1.8 Enabling Policy Logging

Policy logging is expensive; it uses processing time and disk space. In a production environment, you should enable it only under the following types of conditions:

- You have created a new policy and need to verify its functionality.
- You are troubleshooting a policy that is not behaving as expected.

To gather troubleshooting information, you should enable the *File Logging* and *Echo To Console* options in the Identity Server configuration and set the *Component File Logger Levels* for *Application* to at least *info*. Then you must update the Identity Server configuration and restart any Access Gateway Embedded Service Providers, so that the Embedded Service Providers read the logging options. See "Configuring Component Logging" in the *NetIQ Access Manager 3.2 Identity Server Guide*. When you have solved the problem, you should disable these options.

The log file on the component that executed the policy is where you should look for logging information. For example, if you have an Access Gateway: Authorization error, look at the log on the Access Gateway that executed the policy.

For additional policy troubleshooting procedures, see Chapter 7, "Troubleshooting Access Manager Policies," on page 161.

2 Creating Role Policies

This section describes the following topics for Identity Server roles.

- Section 2.1, "Understanding RBAC in Access Manager," on page 23
- Section 2.2, "Creating Roles," on page 27
- Section 2.3, "Example Role Policies," on page 46
- Section 2.4, "Creating Access Manager Roles in an Existing Role-Based Policy System," on page 52
- Section 2.5, "Mapping Roles between Trusted Providers," on page 61
- Section 2.6, "Enabling and Disabling Role Policies," on page 63
- Section 2.7, "Importing and Exporting Role Policies," on page 63

2.1 Understanding RBAC in Access Manager

Role-based access control (RBAC) provides a convenient way to assign a user to a particular job function or set of permissions within an enterprise, in order to control access. As an administrator, you probably have defined a set of roles for your needs. Your roles might include Employee, Student, Administrator, Manager, and so on. You might have Web resources that you want available to all employees, or only to managers, as shown in Figure 2-1.

Figure 2-1 *Traditional RBAC*

Access Manager supports core RBAC functionality by providing user role mapping and the mapping of roles to resource rights and permissions. User role mapping is a primary function of a Role policy. Role mapping to resource rights is accomplished through Authorization policies and role settings in J2EE and SSL VPN environments. When creating a role, you assign users to the role, based on attributes of their identities. You also specify the constraints to place on the role.

Figure 2-2 *RBAC Using a Policy*

As shown in Figure 2-2, during user authentication, the system checks the existing Role policy to determine which roles that a user must be assigned to. After authentication, assigned roles can be used as evaluated conditions of an Authorization policy.

Java applications and Web server applications can also be configured to use roles for access control. For these applications you can use Access Manager to assign the users to the required roles. You can then use the J2EE agent to forward the user's assigned roles to the Java application, or use Access Gateway Identity Injection policies to inject the assigned roles into the HTTP header that is sent to the Web server.

The following examples describe ways to use roles in Access Manager:

- Section 2.1.1, "Assigning All Authenticated Users to a Role," on page 24
- Section 2.1.2, "Using a Role to Create an Authentication Policy," on page 25
- Section 2.1.3, "Using Prioritized Rules in an Authorization Policy," on page 27

2.1.1 Assigning All Authenticated Users to a Role

The system assigns users to roles when they authenticate. The following example illustrates a Role policy that creates an Employee role. All authenticated users are assigned to the role of Employee, because it does not include any conditions (see "Creating an Employee Role" on page 47).

Figure 2-3 *Employee Role Policy*

Role assignment audit events can be created during authentication to the Identity Server. You enabled this on the Logging page in the Identity Server configuration when you enable the *Login Provided* or *Login Consumed* options.

2.1.2 Using a Role to Create an Authentication Policy

The simplest implementation of RBAC policies is to include roles as evaluated conditions when creating Authorization policies.

Suppose you belong to a company of 300 employees, and ten of them are managers. You can assign all employees to an Employee role, and make it a condition of an Authorization policy with no restrictions. Such a policy would permit access to Web resources intended for all employees, as shown in the following example:

Figure 2-4 *Employee Authorization Policy*

For more sensitive Web resources intended only for managers, you might create a role called Manager. (See "Creating a Manager Role" on page 48). The Manager role might be a condition of an Authorization policy that denies access to any employee that has not been assigned to the Manager role when the user authenticated. The following example illustrates this. Notice that the operand for the governing condition logic is set to `If Not`.

Figure 2-5 *Manager Authorization Policy*

After you have created the Authorization policies, you need to assign the policies to the resources they were designed to protect.

See "Assigning an Authorization Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide* and "Assigning a Web Authorization Policy to the Resource" or "Assigning an Enterprise JavaBeans Authorization Policy to a Resource" in the *NetIQ Access Manager 3.2 J2EE Agent Guide*.

2.1.3 Using Prioritized Rules in an Authorization Policy

In another policy example, you might create an Authorization policy for the Sales Department and set up a list of rules that evaluate whether a user has been assigned to one of the roles associated with the department, and then deny access if the user has not been assigned to any of them.

The following image illustrates this scenario:

Figure 2-6 *Authorization Policy with Multiple Rules*

In this example, you specify a first-priority rule with a condition that allows access if a user has been assigned to the role of Sales Representative. You add rules for users assigned to the a role of Sales Manager, Sales Vice President, and so on. You then create a lowest-priority rule that contains no conditions, and an action of Deny. This policy denies any user who has not been assigned a Sales department role. When users do not meet the conditions of the rules, the user is denied access by the lowest-priority rule.

For more information on using roles in Authorization policies, see Chapter 3, "Creating Authorization Policies," on page 65.

2.2 Creating Roles

To implement RBAC, you must first define all of the roles within your organization and the permissions attached to each role. A collection of users requiring the same access can be assigned to a single role. Each user can also be assigned to one or more roles and receive the collective rights associated with the assigned roles. A role policy consists of one or more rules, and each rule consists of one or more conditions and an action.

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New*.
3. Specify a name for the policy, then select *Identity Server: Roles* for the type of policy.
4. Fill in the following fields:

   Description: (Optional) Describe the purpose of this rule. If your role policy contains multiple rules, use the description to identify the purpose of each rule.

Priority: Specify the order in which a rule is applied in the policy, when the policy has multiple rules. The highest priority is 1 and 10 is the lowest.

5 To create a condition for a policy rule, click *New* in the *Condition Group 1* section, then select one of the following:

- **Authenticating IDP:** Specifies the identity provider that authenticated the current user. To use this condition, you must have set up a trusted relationship with more than one identity provider. For configuration information, see Authenticating IDP Condition.
- **Authentication Contract:** Specifies the contract used to authenticate the current user. The selections in this list are defined in the Identity Server configuration. For configuration information, see Authentication Contract Condition.
- **Authentication Method:** Specifies the method used to authenticate the current user. For configuration information, see Authentication Method Condition.
- **Authentication Type:** Compares a selected authentication type to the authentication types used to authenticate the current user. For configuration information, see Authentication Type Condition.
- **Credential Profile:** Requires the user to use the specified credential for authentication. Only values used at authentication time are available for this comparison. For configuration information, see Credential Profile Condition.
- **LDAP Group:** Specifies a group in which the authenticating user is evaluated for membership. For configuration information, see LDAP Group Condition.
- **LDAP OU:** Specifies an OU against which the authenticating user's container is evaluated for containment. For configuration information, see LDAP OU Condition.
- **LDAP Attribute:** Specifies an attribute from the user object of an authenticated user. By default, the selection values include those defined for the InetOrgPerson class. For configuration information, see LDAP Attribute Condition.
- **Liberty User Profile:** Specifies any one of a number of data values that have been mapped to a Liberty Profile attribute. For configuration information, see Liberty User Profile Condition.
- **Roles from Identity Provider:** Specifies a role that has been assigned to the user by an identity provider. For configuration information, see Roles from Identity Provider Condition.
- **User Store:** Compares a selected user store to the user store where the current user is authenticated. For configuration information, see User Store Condition.
- **Condition Extension:** (Conditional) If you have loaded and configured a role condition extension, this option specifies a condition that is evaluated by an outside source. See the documentation that came with the extension for information about what is evaluated.
- **Data Extension:** (Conditional) If you have loaded and configured a role data extension, this option specifies the value that the extension retrieves. You can then select to compare this value with an LDAP attribute, a Liberty User Profile attribute, a Data Entry Field, or another Data Extension. For more information, see the documentation that came with the extension.

NOTE: To improve the policy's performance, configure the LDAP Attributes, Credential Profile, and Liberty User Profile attributes to be sent with authentication. For more information, see "Configuring the Attributes Sent with Authentication"in the *NetIQ Access Manager 3.2 Identity Server Guide*.

---

6 (Conditional) To add multiple conditions, repeat Step 5.

For more information on using multiple conditions in a rule, see Section 2.2.2, "Using Multiple Conditions," on page 42.

7 In the *Actions* section, select one of the following:

- **Activate Role:** Select this option to specify a name for the role. If you are creating a role that needs to be injected into an HTTP header, use the capitalization format that the Web server expects.
- **Activate Selected Role:** Select this option to obtain the role value from an external source.

For more information about specifying a role or roles to activate, see Section 2.2.3, "Selecting an Action," on page 44.

8 Click *OK* twice.

9 Click *Apply Changes*.

10 To enable the role for an Identity Server configuration, see Section 2.6, "Enabling and Disabling Role Policies," on page 63.

2.2.1 Selecting Conditions

You create a role by selecting the appropriate conditions that qualify a user to be assigned to a role, as shown in the following image:

Figure 2-7 *Role Policy Conditions*

The following sections describe the conditions available for a Role policy:

- "Authenticating IDP Condition" on page 30
- "Authentication Contract Condition" on page 31
- "Authentication Method Condition" on page 33
- "Authentication Type Condition" on page 34
- "Credential Profile Condition" on page 35

- "LDAP Group Condition" on page 37
- "LDAP OU Condition" on page 38
- "LDAP Attribute Condition" on page 39
- "Liberty User Profile Condition" on page 39
- "Roles from Identity Provider Condition" on page 40
- "User Store Condition" on page 41
- "Condition Extension" on page 42
- "Data Extension" on page 42

Authenticating IDP Condition

The Authenticating IDP condition allows you to assign a role based on the identity provider that authenticated the current user. To use this condition, you must have set up a trusted relationship with more than one identity provider. See "Configuring SAML and Liberty Trusted Providers" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

The most common way to use this condition is when you have a service provider that has been configured to trust two identity providers and you want to assign a role based on which identity provider authenticated the user. To configure such a policy:

- Set the Authenticating IDP field to *[Current]*
- Set the *Value* field to Authenticating IDP
- Select the name of an identity provider

For the condition to evaluate to True, the identity provider specified in the policy must be the one that the user selected for authentication.

Comparison: Specify how the contract is compared to the data in the *Value* field. Select either a string comparison or a regular expression:

- **Comparison: String:** Specifies that you want the values compared as strings and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the Authenticating IDP value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the Authenticating IDP value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the Authenticating IDP value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments

Dot All
Multi-Line
Unicode
Unix Lines

For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value you want to compare with the Authenticating IDP value. If you select a static value for the Authenticating IDP value, select *Authenticating IDP* and *Current*. If you select *Current* for the Authenticating IDP value, select *Authenticating IDP*, then select the name of an identity provider.

Other value types are possible if you selected *Current* for the Authenticating IDP value. Your policy requirements determine whether they are useful.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

Authentication Contract Condition

The Authentication Contract allows you to assign a role based on the contract the user used for authentication. The Identity Server has the following default contracts:

Name	URI
Name/Password - Basic	basic/name/password/uri
Name/Password - Form	name/password/uri
Secure Name/Password - Basic	secure/basic/name/password/uri
Secure Name/Password - Form	secure/name/password/uri

To configure other contracts for your system, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts*.

The most common way to use this condition is to select *[Current]* for the *Authentication Contract* field and to select *Authentication Contract* and the name of a contract for the *Value* field.

To specify an Authentication Contract condition, fill in the following fields:

Authentication Contract: To compare the contract that the user used with a static value, select *Current*. To compare a static value with what the user used, select a contract from the list.

If you have created more than one Identity Server configuration, select the configuration, then select the contract. The name of the contract is displayed. When you select this name, the configurations that contain a definition for this contract are highlighted.

For example, the following policy has selected *Name/Password - Basic* as the contract.

Two Identity Server configurations have been defined (idp-43.amlab.net and idp-51.amlab.net). Both configurations are highlighted because *Name/Password - Basic* is a contract that is automatically defined for all Identity Server configurations.

If the contract you are selecting for a condition is a contract with ORed credentials, you need to use multiple conditions to set up a rule. See "Creating a Rule for a Contract with ORed Credentials" on page 50.

Comparison: Specify how the contract is compared to the data in the *Value* field. Select either a string comparison or a regular expression:

- **Comparison: String:** Specifies that you want the values compared as strings and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the Authentication Contract value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the Authentication Contract value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the Authentication Contract value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value you want to compare with the Authentication Contract value. If you select a static value for the Authentication Contract value, select *Authentication Contract* and *Current*. If you select *Current* for the Authentication Contract value, select *Authentication Contract*, then select the name of a contract.

Other value types are possible if you selected *Current* for the Authentication Contract value. For example:

- You can select *Data Entry Field*. The value specified in the text box must be the URI of the contract for the conditions to match. For a list of these values, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts*.
- If you have defined a Liberty User Profile attribute for URI of the authentication contract, you can select *Liberty User Profile,* then select the attribute.
- If you have defined an LDAP attribute for URI of the authentication contract, you can select *LDAP Attribute,* then select the attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

Authentication Method Condition

The Authentication Method allows you to assign a role based on the method the user used for authentication.

Authentication Method: To compare the method that the user used with a static value, select *Current*. To compare a static value with what the user used, select a method from the list.

If you have created more than one Identity Server configuration, select the configuration, then select the method. The name of the method is displayed. When you select this name, the configurations that contain a definition for this method are highlighted.

Comparison: Specify how the method is compared to the data in the *Value* field. Select either a string comparison or a regular expression:

- **Comparison: String:** Specifies that you want the values compared as strings and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the Authentication Method value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the Authentication Method value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the Authentication Method value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence

Case Insensitive
Comments
Dot All
Multi-Line
Unicode
Unix Lines

For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value you want to compare with the Authentication Method value. If you select a static value for the Authentication Method value, select *Authentication Method* and *Current*. If you select *Current* for the Authentication Method value, select *Authentication Method*, then select the name of a method.

Other value types are possible if you selected *Current* for the Authentication Method value. Your policy requirements determine whether they are useful.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

Authentication Type Condition

The Authentication Type condition allows you to assign a role based on the authentication types used to authenticate the current user. The [Current] selection represents the current set of authentication types used to authenticate the user. The other selections represent specific authentication types that can be used to compare with [Current]. The Authentication Type condition returns true if the selected Authentication Type is contained in the set of Authentication Types for [Current]. For example, if the current user was required to satisfy the Authentication Types of Basic and SmartCard, then a selected Authentication Type of either Basic or SmartCard would match.

Authentication Type: To compare the type that the user used with a static value, select *Current*. To compare a static value with what the user used, select a type from the list.

Comparison: Specify how the type is compared to the data in the *Value* field. Select either a string comparison or a regular expression:

- **Comparison: String:** Specifies that you want the values compared as strings and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the Authentication Type value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the Authentication Type value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the Authentication Type value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

Canonical Equivalence
Case Insensitive
Comments
Dot All
Multi-Line
Unicode
Unix Lines

For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value you want to compare with the Authentication Type value. If you select a static value for the Authentication Type value, select *Authentication Type* and *Current*. If you select *Current* for the Authentication Type value, select *Authentication Type*, then select a type.

Other value types are possible if you selected *Current* for the Authentication Type value. Your policy requirements determine whether they are useful.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

Credential Profile Condition

The Credential Profile condition allows you to assign a role based on the credentials the user entered when authenticating to the system. Only values used at authentication time are available for this comparison.

To set up the matching for this condition, fill in the following fields:

Credential Profile: Specify the type of credential your users are using for authentication. If you have created a custom contract that uses a credential other than the ones listed below, do not use the Credential Profile as a Role condition.

- **LDAP Credentials:** If you prompt the user for a username, select this option, then select *LDAP User Name* (the cn of the user) or *LDAP User DN* (the fully distinguished name of the user), or *LDAP Password*.

  The default contracts assign the cn attribute to the Credential Profile. If your user store is an Active Directory server, the SAMAccountName attribute is used for the username and stored in the cn field of the LDAP Credential Profile.
- **X509 Credentials:** If you prompt the user for a certificate, select this option, then select one of the following:
  - **X509 Public Certificate Subject:** Retrieves the subject field from the certificate, which can match the DN of the user, depending upon who issued the certificate.
  - **X509 Public Certificate Issuer:** Retrieves the issuer field from the certificate, which is the name of the certificate authority (CA) that issued the certificate.
  - **X509 Public Certificate:** Retrieves the entire certificate, Base64 encoded.
  - **X509 Serial Number:** Retrieves the serial number of the certificate.
- **SAML Credential:** If your users authenticate with a SAML assertion, select this option.

Comparison: Select one of the following types:

- **Comparison: String:** Specifies that you want the values compared as strings and indicates how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the Credential Profile value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the Credential Profile value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the Credential Profile value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the second value for the comparison. Select one of the following data types:

- **LDAP Attribute:** If you have an LDAP attribute that corresponds to the Credential Profile you have specified, select this option and the attribute.
- **Liberty User Profile:** If you have a Liberty User Profile attribute that corresponds to the Credential Profile you have specified, select this option and the attribute.
- **Data Entry Field:** Specify the string you want matched. Be aware of the following requirements:
  - If you selected *LDAP User DN* as the credential, you need to specify the DN of the user in the *Value* text box. If the comparison type is set to *Contains Substring*, you can match a group of users by specifying a common object that is part of their DNs, for example `ou=sales`.
  - If you selected *X509 Public Certificate Subject* as the credential, you need to specify all elements of the Subject Name of the certificate in the *Value* text box. Separate the elements with a comma and a space, for example, `o=novell, ou=sales`. If the comparison type is set to *Contains Substring*, you can match a group of certificates by specifying a name that is part of the Subject Name, for example `ou=sales`.

Other values are possible. Your policy requirements determine whether they are useful.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

LDAP Group Condition

The LDAP Group condition allows you to assign a role based on whether the authenticating user is a member of a group. The value, an LDAP DN, must be a fully distinguished name of a group.

LDAP Group: Select *[Current]*.

Comparison: Specify how you want the values compared. Select one of the following:

- **LDAP Group: Is Member of:** Specifies that you want the condition to determine whether the user is member of a specified group.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: If you selected *Regular Expression: Matches* as the comparison type, select one or more of the following:

Canonical Equivalence
Case Insensitive
Comments
Dot All
Multi-Line
Unicode
Unix Lines

For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the second value for the comparison. If you select *LDAP Group > Name of Identity Server Configuration > User Store Name*, you can browse to the name of the LDAP group.

If you have more than 250 groups in your tree, you are prompted to enter an LDAP query string. In the text box, you need to add only the <strFilter> value for the query. For example:

<strFilter> Value	Description
admin*	Returns all groups that begin with admin, such as adminPR, adminBG, and adminWTH.
*test	Returns all groups that end with test, such as doctest, softtest, and securtest.
low	Returns all groups that have “low” in the name, such as low, yellow, and clowns.

For more information about the <strFilter> parameter, see RFC 2254 “LDAP Search Filter.”

If you select *Data Entry Field* as the value, you can specify the DN of the group in the text field. For example:

```
cn=managers,cn=users,dc=bcf2,dc=provo,dc=novell,dc=com
```

```
cn=manager,o=novell
```

Other values are possible. Your policy requirements determine whether they are useful.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

LDAP OU Condition

The LDAP OU condition allows you to assign a role based on a comparison of the DN of an OU against the DN of the authenticated user. If the user's DN contains the OU, the condition matches.

LDAP OU: Select *[Current]*.

Comparison: Specify how you want the values compared. Select one of the following:

- **Contains:** Specifies that you want the condition to determine whether the user is contained by a specified organizational unit.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type.

- **Contains:** Select whether the user must be contained in the specified OU (*One Level)* or whether the user can be contained in the specified OU or a child container (*Subtree)*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the second value for the comparison. If you select *LDAP OU > Name of Identity Server Configuration > User Store Name*, you can browse to the name of the OU.

If you have more than 250 OUs defined in your tree, you are prompted to enter an LDAP query string. In the text box, you need to add only the <strFilter> value for the query. For example:

<strFilter> Value	Description
admin*	Returns all OUs that begin with admin, such as adminPR, adminBG, and adminWTH.
*test	Returns all OUs that end with test, such as doctest, softtest, and securtest.
low	Returns all OUs that have "low" in the name, such as low, yellow, and clowns.

For more information about the <strFilter> parameter, see RFC 2254 "LDAP Search Filter."

If you select *Data Entry Field,* you can specify the DN of the OU in the text field. For example:

```
cn=users,dc=bcf2,dc=provo,dc=novell,dc=com
```

```
ou=users,o=novell
```

If you have defined a Liberty User Profile or an LDAP attribute for the OU you want to match, select this option, then select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

LDAP Attribute Condition

The LDAP Attribute condition allows you to assign a role based on a value in an LDAP attribute defined for the inetOrgPerson class or any other LDAP attribute you have added. You can have the user's attribute value retrieved from your LDAP directory and compared to a value of the following type:

- Roles from an identity provider
- Authenticating IDP or user store
- Authentication contract, method, or type
- Credential profile
- LDAP attribute, OU, or group
- Liberty User Profile attribute
- Static value in a data entry field

To set up the matching for this condition, fill in the following fields:

LDAP Attribute: Specify the LDAP attribute you want to use in the comparison. Select from the listed LDAP attributes. To add an attribute that isn't in the list, click *New LDAP Attribute*, then specify the name of the attribute.

Comparison: Specify how you want the values compared. All data types are available. Select one that matches the value type of your attribute.

Mode: Select the mode, if available, that matches the comparison type. For example, if you select to compare the values as strings, you can select either a *Case Sensitive* mode or a *Case Insensitive* mode.

Value: Specify the second value for the comparison. All data types are available. For example, you can select to compare the value of one LDAP attribute to the value of another LDAP attribute. Only you can determine if such a comparison is meaningful.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

Liberty User Profile Condition

The Liberty User Profile condition allows you to assign a role based on a value in a Liberty User Profile attribute. The Liberty attributes must be enabled before you can use them in policies (click *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*, then enable one or more of the following: *Employee Profile* or *Personal Profile*).

These attributes can be mapped to LDAP attributes (click *Identity Servers* > *Edit* > *Liberty* > *LDAP Attribute Mapping*). When mapped, the actual value comes from your user store. If you are using multiple user stores with different LDAP schemas, mapping similar attributes to the same Liberty User Profile attribute allows you to create one policy with the Liberty User Profile attribute rather than multiple policies for each LDAP attribute.

The selected attribute is compared to a value of the following type:

- Roles from an identity provider
- Authenticating IDP or user store
- Authentication contract, method, or type
- Credential profile
- LDAP attribute, OU, or group
- Liberty User Profile attribute
- Static value in a data entry field

To set up the matching for this condition, fill in the following fields:

Liberty User Profile: Select the Liberty User Profile attribute. These attributes are organized into three main groups: Custom Profile, Corporate Employment Identity, and Entire Personal Identity. By default, the Common Last Name attribute for Liberty User Profile is mapped to the sn attribute for LDAP. To select this attribute for comparison, click *Entire Personal Identity* > *Entire Common Name* > *Common Analyzed Name* > *Common Last Name*.

Comparison: Select the comparison type that matches the data type of the selected attribute and the value.

Mode: Select the mode, if available, that matches the data type. For example, if you select to compare the values as strings, you can select either a *Case Sensitive* mode or a *Case Insensitive* mode.

Value: Select one of the values that is available from the current request or select *Data Entry Field* to enter a static value. The static value that you can enter depends on the comparison type you selected.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

Roles from Identity Provider Condition

The Roles from Identity Provider condition allows you to assign a role based on a role assigned by another identity provider (Liberty, SAML 2.0, WS Federation). You configure the condition to match the role sent by the identity provider, then set the action to assign a new role.

This condition uses the mapped attribute All Roles. All roles that are assigned to the user can be mapped to attributes and assigned to a trusted identity provider. For information about enabling All Roles, see "Selecting Attributes for a Trusted Provider" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

For an example of how to use Roles from Identity Provider to create a Role policy, see Section 2.5, "Mapping Roles between Trusted Providers," on page 61. For an example that explains all the configuration procedures required for sharing roles, see "Sharing Roles" in the *NetIQ Access Manager 3.2 Setup Guide*.

To configure a Roles from Identity Provider condition, fill in the following fields:

Roles from Identity Provider: If you have configured your system for multiple identity providers, select the identity provider. If you have only one, it is selected.

Comparison: Select one of the following types:

- **Comparison: String:** Specifies that you want the values compared as strings, and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the Roles from Identity Provider value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the Roles from Identity Provider value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the Roles from Identity Provider value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Select *Data Entry Field*, then specify the name of an identity provider role. Other value types are possible. Your policy requirements determine whether they are useful

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

User Store Condition

The User Store condition allows you to assign a role based on the user store that was used to authenticate the current user. The [Current] selection represents the user store from which the user was authenticated. The other selections represent all of the configured user stores that can be used to compare with [Current].

For example, if the configured user stores are eDir1 and AD1 and the current user is authenticated from eDir1, then a selected user store of eDir1 would match and a selected user store of AD1 would not match.

User Store: To compare the user store that the user used for authentication with a static value, select *Current*. To compare a static value with what the user used, select a user store from the list.

If you have created more than one Identity Server configuration, select the configuration, then select the user store. The name of the user store is displayed.

Comparison: Specify how the user store is compared to the data in the *Value* field. Select either a string comparison or a regular expression:

- **Comparison: String:** Specifies that you want the values compared as strings and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the User Store value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the User Store value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the User Store value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Value: Specify the value you want to compare with the User Store value. If you select a static value for the User Store value, select *User Store* and *Current*. If you select *Current* for the User Store value, select *User Store*, then select the name of a user store.

If you have created more than one Identity Server configuration, select the configuration, then select the user store. The name of the user store is displayed.

Other value types are possible if you selected *Current* for the User Store value. Your policy requirements determine whether they are useful.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

Condition Extension

If you have loaded and configured a role condition extension, this option specifies a condition that is evaluated by an outside source. See the documentation that came with the extension for information about what is evaluated.

Data Extension

If you have loaded and configured a role data extension, this option specifies the value that the extension retrieves. You can then select to compare this value with an LDAP attribute, a Liberty User Profile attribute, a Data Entry Field, or another Data Extension. For more information, see the documentation that came with the extension.

2.2.2 Using Multiple Conditions

The *Condition structure* field controls how conditions within a condition group interact with each other and how condition groups interact with each other. Select one of the following:

- "AND Conditions, OR groups" on page 43
- "OR Conditions, AND groups" on page 43

The following sections explain how to configure the condition groups and conditions to interact with each other:

- "Using the Not Options" on page 44
- "Adding Multiple Conditions" on page 44
- "Adding New Condition Groups" on page 44
- "Disabling Conditions and Condition Groups" on page 44

AND Conditions, OR groups

If the conditions are ANDed, the user must meet all the conditions in a condition group to match the profile. If the condition groups are ORed, the user must meet all of the conditions of one group to match the profile. This option allows you to set up two or more profiles into which a user could fit and be considered a match. For example, suppose you create the following Permit rule.

The first condition group contains the following conditions:

1. The user's department must be Engineering.
2. The request must come on a weekday.

The second condition group contains the following conditions:

1. The user's department must be Information Services and Technology (IS&T).
2. The request must come on a weekend.

With this rule, the engineers who match the first condition group have access to the resource during the week, and the IS&T users who match the second condition group have access to the resource on the weekend.

OR Conditions, AND groups

If the conditions are ORed, the user must meet at least one condition in the condition group to match the profile. If the conditions groups are ANDed, the user must meet at least one condition in each condition group to match the profile. For example, suppose you created the following Permit rule:

The first condition group contains the following conditions:

1. The user's department is Engineering.
2. The user's department is Sales.

The second condition group contains the following conditions:

1. The user has been assigned the Party Planning role.
2. The user has been assigned the Vice President role.

With this rule, the Vice Presidents of both the Engineering and Sales departments can access the resource, and the users from the Engineering and Sales department who have been assigned to the Party Planning role can access the resource.

Using the Not Options

At the top of each condition group, there is an option that allows you to control whether the user must match the conditions to match the profile or whether the user matches the profile if the user doesn't match any of the conditions. Depending upon your selection for the Condition structure, you can select from the following:

- If/If Not
- Or/Or Not
- And/And Not

Conditions also have similar Not options, so that a user can match a condition by not matching the specified value.

Adding Multiple Conditions

To add another condition to a condition group, click *New*, then select a condition. To copy an existing condition, click the *Copy Condition* icon. New conditions are always added to the end of the condition group. Use the *Move* buttons to order the conditions in the condition group.

Adding New Condition Groups

To add another condition group to the rule, click *Append New Group*. To copy the existing condition group, click the *Copy Group* icon. New condition groups are always added to the end to the Conditions section. Use the *Move* buttons to order the condition groups.

Disabling Conditions and Condition Groups

Condition groups and conditions within them can be disabled by clicking the Enabled check mark, which changes the icon to the *Disabled* icon.

You usually disable a condition or condition group when testing a new rule, and if you decide the condition or condition group is not needed, you can then use the *Delete* button to delete the condition or condition group from the rule. Use the *Move* buttons by the *Delete* button to move a condition up or down within its group. Condition groups also have *Move* buttons.

2.2.3 Selecting an Action

The policy action specifies the role to which the user is assigned. Roles are activated at the time the role policy is evaluated. Select one of the following actions:

- "Activate Role" on page 44
- "Activate Selected Role" on page 45

Activate Role

Select *Activate Role* when you want to specify a name for the role. If you are creating a role that needs to injected into an HTTP header, use the same capitalization format as the Web server expects. For example, if the Web server expects an Employee role with an initial capital, name your role Employee.

Figure 2-8 show how to assign the role of Employee to a policy.

Figure 2-8 *Assigning a Role*

To use the same conditions to activate multiple roles, select *Activate Role* for each role you want to specify.

Activate Selected Role

Select *Activate Selected Role* when you want to obtain the role value from an external source. Select one of the following:

- **LDAP Attribute:** If you have an LDAP attribute that is a role, select the attribute from the list. If the attribute is not in the list, select *New LDAP Attribute* to add it to the list.
- **LDAP Group:** Activates a role based on an LDAP Group attribute. Select either [Current] or browse to the DN of the group by selecting the Identity Server and User Store. The value for this option is the DN of the group. If you select [Current], the value can be a list of the groups the user belongs to. The [Current] value makes the DN of each group in the attribute into a role.

  If you select to browse to the DN of the group and you have more than 250 groups in your tree, you are prompted to enter an LDAP query string. In the text box, you need to add only the <strFilter> value for the query. For example:

<strFilter> Value	Description
admin*	Returns all groups that begin with admin, such as adminPR, adminBG, and adminWTH.
*test	Returns all groups that end with test, such as doctest, softtest, and securtest.
low	Returns all groups that have “low” in the name, such as low, yellow, and clowns.

  For more information about the <strFilter> parameter, see RFC 2254 “LDAP Search Filter.”

  This action does not query all the static and dynamic groups on the LDAP server to see if the user belongs to them, but uses the user’s group membership attribute to create the list. If you want to use this longer query, you need to create a policy extension. For a sample extension that does this, see Novell Access Manager Developer Tools and Examples (http://developer.novell.com/wiki/index.php/Novell_Access_Manager_Developer_Tools_and_Examples).

- **LDAP OU:** Activates a role based on the Organizational Unit in the user’s DN. Select either [Current] or browse to the DN of the OU by selecting the Identity Server and User Store. The value for this option is the DN of the OU.

  If you select to browse to the DN of the OU and you have more than 250 OUs defined in your tree, you are prompted to enter an LDAP query string. In the text box, you need to add only the <strFilter> value for the query. For example:

<strFilter> Value	Description
admin*	Returns all OUs that begin with admin, such as adminPR, adminBG, and adminWTH.
*test	Returns all OUs that end with test, such as doctest, softtest, and securtest.
low	Returns all OUs that have “low” in the name, such as low, yellow, and clowns.

For more information about the <strFilter> parameter, see RFC 2254 “LDAP Search Filter.”

- **Liberty User Profile:** If you have a Liberty attribute that is a role, select the attribute from the list.
- **Data Extension:** If you have created a data extension that calculates a set of roles, select the extension. For information on creating such an extension, see Novell Access Manager Developer Tools and Examples (http://developer.novell.com/wiki/index.php/Novell_Access_Manager_Developer_Tools_and_Examples).

If the source contains multiple values, select the format that is used to separate the values.

If the value is a distinguished name, select the format of the DN.

Figure 2-9 shows how to assign an LDAP Group, cn=DocGroup,o=novell, as a role.

Figure 2-9 *Activating a Role from an External Source*

To use the same conditions to activate multiple roles from different sources, select *Activate Selected Role* for each role you want to activate.

2.3 Example Role Policies

The following examples describe how to create a general Employee role, a restrictive Manager role, and a role from a contract with ORed credentials. These roles can be used by the Access Gateway in Identity Injection policies and by the Access Gateway and the J2EE Agent in Authorization policies.

- Section 2.3.1, “Creating an Employee Role,” on page 47
- Section 2.3.2, “Creating a Manager Role,” on page 48
- Section 2.3.3, “Creating a Rule for a Contract with ORed Credentials,” on page 50

2.3.1 Creating an Employee Role

The following role policy creates an Employee role. Because the role does not include conditions, all authenticated users are assigned to this role when they log in. This role can then be used to grant access to resources to all users in your user stores.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Roles* > *Manage Policies.*
2. On the Policies page, click *New*.

3. Select a policy type of *Identity Server: Roles* and specify a display name, such as Employee.
4. Click *OK*.
5. On the Edit Policy page, specify a description in the *Description* field.

   It is important to use this field to keep track of your roles and policies. The policy feature is powerful, and your setup can be as large and complex as you want it to be, with a potentially unlimited number of conditions and choices. This description is useful to help keep track of various role and policy configurations.
6. Make sure the *Condition Group 1* section has no conditions, so that all users who authenticate match the condition.

7 In the *Actions* section, click *New* > *Activate Role*.

8 In the *Activate Role* box, type `Employee`, then click *OK*.

If this role needs to match the name of a role required by a Java or Web application, ensure that the case of the name matches the application's name.

9 On the Rule List page, click *OK*.

10 On the Policies page, click *Apply Changes*, then click *Close*.

11 On the Role Policy page, select the Employee role, then click *Enable.*

12 Click *OK*, then update the Identity Server.

The Identity Server configuration must be updated after you enable a role.

13 To create a Manager role, continue with "Creating a Manager Role" on page 48.

2.3.2 Creating a Manager Role

Because the Manager role is restrictive, role policy conditions must be specified. The Manager role is assigned only to the users who meet the conditions.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Roles* > *Manage Policies.*

2 On the Policies page, click *New*.

3 Select a policy type of *Identity Server: Roles* and specify a display name (for this example, Manager.)

4 Click *OK*.

5 In the *Conditions* section, click *New* > *Liberty User Profile*.

6 In *Condition Group 1*, select the conditions the user must meet:

Liberty User Profile: Select *Entire Personal Identity* > *Entire Common Name* > *Common Analyzed Name* > *Common Last Name.*

If these options are not available, you haven't enabled the Liberty attributes. Click *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*, then enable one or more of the following: *Employee Profile* or *Personal Profile.*

Comparison: Select how you want the attribute values to be compared. For the Common Last Name attribute, select *String* > *Equals*.

Mode: Select *Case Insensitive*.

Value: Select *Data Entry Field* and type the person's name in the box (Smith, in this example). This sets up the condition that if the user has the name Smith, his or her role as Manager is activated at authentication.

Result on Condition Error: This sets up the results that are returned if an error occurs while evaluating the condition (for example, the LDAP server goes down). This rule is set up to grant the user the role of Manager if the condition evaluates to *True*. If an error occurs, you do not want random users assigned the role of Manager. Therefore, for this rule, you need to select *False*.

7 In the *Actions* section, click *Activate Role*.

8 In the *Activate Role* box, type `Manager`, then click *OK* twice.

9 On the Policies page, click *Apply Changes*.

10 Click *Close*, select the Manager role, then click *Enable*.

11 Click *OK*, then update the Identity Server.

2.3.3 Creating a Rule for a Contract with ORed Credentials

A contract with ORed credentials allows the user to decide which credentials to use for authenticating. If you are creating a role policy that grants the user the role regardless of which method was used for authentication, you can use such a contract just as you would any other contract in a condition. However, if you want to base the condition on the user using the contract with multiple credentials for authentication and on the user authenticating with a particular credential (password, token, or certificate), you need to create a rule with two conditions: one condition checks for the contract and the second condition checks for the authenticating credential.

If the contract with ORed credentials was named OringContracts, the first condition in the rule should look similar to the following:

Figure 2-10 *Checking for the Contract*

This condition verifies that the user used the OringContracts contract for authentication. The second condition needs to verify the type of credential that was used. To do this, you need to check for the existence of the credential in the Credential Profile. This condition should look similar to the following if you are verifying that the user used a certificate for the credential.

Figure 2-11 *Checking for the Credential*

The policy engine evaluates the above condition to true when the Credential Profile contains a value for the certificate. If the user used another method for authentication, the certificate field is empty, and the policy engine evaluates two null entries to false.

This type of condition works for the LDAP credentials and the X.509 credentials. It does not work for the Radius token, because the Credential Profile does not store the Radius token. You need to use "If Not" logic to verify that the user authenticated with a token. For example, if the OringContracts contract ORed the Radius token class with the Name/Password class, you would know that the user authenticated with a token when the password credential has no value. This type of condition should look similar to the following:

Figure 2-12 *Using "If Not" Logic*

If the Credential Profile contains a value for the password, this condition evaluates to false because of the "And If Not" logic. If the password value in the Credential Profile is empty, this condition evaluates to true, and you know that the user authenticated with a Radius token.

2.4 Creating Access Manager Roles in an Existing Role-Based Policy System

If you have already implemented a role-based administration policy for granting access to print, file, and LDAP resources, you can leverage your role definitions and use Access Manager policies to control access to Web resources. If your role definitions use the following types of LDAP features, you can create Access Manager Role policies that use them:

- Values found in LDAP attributes
- Location of the user objects in the directory tree
- Membership in groups or roles

The Access Manager Role policies that you create for these features can then be used to control access to protected Web resources. You can manually assign the roles by creating role policies with conditions or you can activate roles based on the values in the external source.

- Section 2.4.1, "Activating Roles from External Sources," on page 52
- Section 2.4.2, "Using Conditions to Assign Roles," on page 54

2.4.1 Activating Roles from External Sources

If you have an LDAP attribute, an LDAP group, an LDAP OU, or a Liberty attribute that you are currently using for role assignments, you can have Access Manager read its value and activate roles based on the values. This allows you to use the same roles for Access Manager access as you are using in other parts of your deployment.

When you create this type of Role policy, you do not need to specify any conditions. The policy engine reads the attribute you specify, then assigns roles to users based on the value or values in the attribute. If the user has no value for the attribute, the user is assigned no roles. If the user has a value for the attribute, the user is assigned a role for each value in the attribute.

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New* to create a new policy.
3. Specify a name for the Role policy, select *Identity Server: Roles* for the type, then click *OK*.
4. On the Rule page in the *Actions* section, click *New* > *Activate Selected Role*.
5. For this example, select *LDAP Group*.
6. To select the group you want to use for role assignments, click *Current* > *[Identity Server Name]* > *[User Store Name]* > *[Group Name]*.

   The distinguished name of this group is the Role name that is assigned to the user.
7. Select a *Multi-Value Separator* that is compatible with a distinguished name.

   A comma, which is the default separator, cannot be used because a comma is used to separate the components in a distinguished name. Select any other value, such as #.

Your policy should look similar to the following:

8 Click *OK* twice, then click *Apply Changes*.

9 To enable the role so that it can be used in Authorization and Identity Injection policies, click *Devices* > *Identity Servers* > *Edit* > *Roles*.

10 Select the check box next to the name of the role, then click *Enable*.

11 Click *OK*.

12 Update the Identity Server.

13 (Optional) Verify the name used for the role and the user assigned to it:

13a Enable logging by clicking *Devices* > *Identity Servers* > *Edit* > *Logging*, then set the following values:

File Logging: Select *Enabled*.

Echo To Console: Select this option to enable it.

Application: In the *Component File Logger Levels* section, set to *info*.

13b Click *OK*, then update the Identity Server.

13c Log in to the Identity Server by using the credentials of a user who belongs the LDAP group.

13d View the log file for the Identity Server by clicking *Auditing* > *General Logging*

13e Select the file (for Windows, select the `stdout.log` file; for Linux, select the `catalina.out` file), then click *Download*.

13f Look for two log entries (<amLogEntry>) similar to the following:

```
<amLogEntry> 2009-10-09T21:58:55Z INFO NIDS Application: AM#500199050:
AMDEVICEID#CA50FD51DB1EEE3E: AMAUTHID#213E610199A14CEAF27395A6B35F3162:
IDP RolesPep.evaluate(), policy trace:
   ~~RL~1~~~~Rule Count: 1~~Success(67)
   ~~RU~RuleID_1223587171711~LDAP_Group~DNF~~0:1~~Success(67)
   ~~PA~ActionID_1223588319336~~AddSelectedRoles~cn=Doc~~~Success(0)
   ~~PA~ActionID_1223588319336~~AddSelectedRoles~o=novell~~~Success(0)
   ~~PC~ActionID_1223588319336~~Document=(ou=xpemlPEP,ou=mastercdn,
ou=ContentPublisherContainer,ou=Partition,ou=PartitionsContainer,
ou=VCDN_Root,ou=accessManagerContainer,o=novell:romaContentCollection
XMLDoc),Policy=(LDAP_Group),Rule=(1::RuleID_1223587171711),Action=
(AddSelectedRole::ActionID_1223588319336)~~~~Success(0)
 </amLogEntry>

<amLogEntry> 2009-10-09T21:58:55Z INFO NIDS Application: AM#500105013:
AMDEVICEID#CA50FD51DB1EEE3E: AMAUTHID#213E610199A14CEAF27395A6B35F3162:
Authenticated user cn=jwilson,o=novell in User Store Internal with roles
"cn=Doc,o=novell","authenticated".
</amLogEntry>
```

The first <amLogEntry> entry indicates that the action in the LDAP_Group policy was successfully assigned.

The second entry gives the DN of the user and lists the roles assigned to the user: cn=Doc,o=novell and authenticated.

You can now use the cn=Doc,o=novell role when creating Authorization and Identity Injection policies, which control access to protected Web resources. Roles activated this way do not appear in the list of available roles. You need to use the *Data Entry Field* to manually type in the role name. For more information, see the following:

- Chapter 3, "Creating Authorization Policies," on page 65
- Chapter 4, "Creating Identity Injection Policies," on page 113

2.4.2 Using Conditions to Assign Roles

- "Creating a Role by Using an LDAP Attribute" on page 54
- "Creating a Role by Using the Location of the User Objects" on page 56
- "Creating a Role by Using a Group Membership Attribute" on page 58

Creating a Role by Using an LDAP Attribute

You can assign a user to a role by using a value found in any LDAP attribute in your directory. The following example uses the objectClass attribute because every object in an LDAP directory has an objectClass attribute that contains the object classes to which the object belongs. This attribute contains the name of the object class that was used to create the object as well as the names of the superior object classes of this class. All you need to know is the name of the object class you used to create your users in the LDAP directory. For example, the following instructions create a Role policy for users who were created with the User object class.

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New*.
3. Specify a name for the Role policy, select *Identity Server: Roles* for the type, then click *OK*.
4. In *Condition Group 1*, click *New*, then select *LDAP Attribute*.
5. In *Condition Group 1*, select the conditions the user must meet:

LDAP Attribute: Select the objectClass attribute. If you have not added this attribute, it won't appear in the list. Scroll to the bottom of the list, click *New LDAP Attribute*, specify objectClass for the name, then click *OK*.

If you are using eDirectory™ for your LDAP directory, you need to specify standard LDAP names for the attributes. Access Manager does not support spaces or colons in attribute names.

Comparison: Select how you want the attribute values to be compared. For the objectClass attribute, select *String* > *Contains Substring*.

The objectClass attribute is a multi-valued attribute and, for most objects, contains multiple values. For example in eDirectory, users created with the User object class have User, organizationalPerson, person, ndsLoginProperties, and top as values in the objectClass attribute.

Mode: Select *Case Insensitive*.

Value: Select *Data Entry Field* and specify User as the value.

Result on Condition Error: This sets up the results that are returned if an error occurs while evaluating the condition (for example, the LDAP server goes down). This rule is set up to grant the user the role of UserClass if the condition evaluates to *True*. If an error occurs, you do not want random users assigned the role of UserClass. Therefore, for this rule, you need to select *False*.

6 In the *Actions* section, click *Activate Role*.

7 In the *Activate Role* box, type `UserClass`, then click *OK*.

The name you specify in the box is the role you want assigned to the users who match the condition.

Your rule should look similar to the following:

8 Click *OK* twice, then click *Apply Changes*.

9 To enable the role so that it can be used in Authorization and Identity Injection policies, click *Identity Servers* > *Edit* > *Roles*.

10 Select the check box next to the name of the role, then click *Enable*.

11 Click *OK*.

12 Update the Identity Server.

You can now use this role when creating Authorization and Identity Injection policies, which control access to protected Web resources. For more information, see the following:

- Chapter 3, "Creating Authorization Policies," on page 65
- Chapter 4, "Creating Identity Injection Policies," on page 113

Creating a Role by Using the Location of the User Objects

If you have created your users in specific containers in your LDAP tree, you can use these container objects to assign users to roles. For example, suppose your LDAP tree looks similar to the following tree.

Figure 2-13 *Using an eDirectory Tree for Access Control*

Such a tree organization can be used to control access to resources. The following instructions explain how to create a Role policy for the users created under the Sales container.

1 In the Administration Console, click *Policies* > *Policies*.

2 Select the policy container, then click *New*.

3 Specify a name for the Role policy, select *Identity Server: Roles* for the type, then click *OK*.

4 In *Condition Group 1*, click *New*, and select *LDAP OU* > *[Identity Server Configuration]* > *[User Store]* > *[DN of the OU]*.

The following example illustrates how to make these selections:

Comparison: Select how you want the attribute values to be compared. For LDAP OU, select *Contains*.

Mode: Select *One Level* if all your users are created in ou=Sales. Select *Subtree* if your users are created in various containers under the ou=Sales container.

Value: Select *LDAP OU*, then select *[Current]*.

The DN of the authenticated user is compared with the value specified in LDAP OU. If the DN of the user contains the LDAP OU value, the user matches the condition. For example, if the DN of the user is cn=bsmith,ou=sales,o=novell and the LDAP OU value is ou=sales,o=novell, the user matches the condition. If you selected *Subtree* for the Mode, a user with the following DN also matches the condition: cn=djones,ou=provo,ou=sales,o=novell.

Result on Condition Error: This sets up the results that are returned if an error occurs while evaluating the condition (for example, the LDAP server goes down). This rule is set up to grant the user the role of Sales if the condition evaluates to *True*. If an error occurs, you do not want random users assigned the role of Sales. Therefore, for this rule, you need to select *False*.

5 In the *Actions* section, click *Activate Role*.

6 In the *Activate Role* box, type `Sales`, then click *OK*.

The name you specify in the box is the role you want assigned to the users who match the condition.

Your rule should look similar to the following:

7 Click *OK* twice, then click *Apply Changes*.

8 To enable the role so that it can be used in Authorization and Identity Injection policies, click *Devices* > *Identity Servers* > *Edit* > *Roles*.

9 Select the check box next to the name of the role, then click *Enable*.

10 Click *OK*.

11 Update the Identity Server.

You can now use this role when creating Authorization and Identity Injection policies, which control access to protected Web resources. For more information, see the following:

- Chapter 3, "Creating Authorization Policies," on page 65
- Chapter 4, "Creating Identity Injection Policies," on page 113

Creating a Role by Using a Group Membership Attribute

If you have created an LDAP group and assigned users to the group, you can use group membership to assign a role to the user. For example, you might have created a first-level managers group and made all your first-level managers members of this group. You can then have other groups for your upper-level managers. You can create a Role policy that assigns the user a role if the user is a member of a specific group. The Role policy can then be used in an Authorization or Identity Injection policy to protect a Web resource.

1 In the Administration Console, click *Policies* > *Policies*.

2 Select the policy container, then click *New*.

3 Specify a name for the Role policy, select *Identity Server: Roles* for the type, then click *OK*.

4 In *Condition Group 1*, click *New*, then select *LDAP Group*.

5 In *Condition Group 1*, select the conditions the user must meet:

LDAP Group: Select the Identity Server Configuration, the user store, then the Group.

The following figure illustrates this selection process.

Comparison: Select how you want the attribute values to be compared. For LDAP Group, select *Is Member of*.

Value: Select *LDAP Group*, then select *[Current]*.

The DN of the authenticated user is compared with the members of the LDAP Group. If the DN of the user matches one of the members, the user matches the condition.

Result on Condition Error: This sets up the results that are returned if an error occurs while evaluating the condition (for example, the LDAP server goes down). This rule is set up to grant the user the role of ManagersGroup if the condition evaluates to *True*. If an error occurs, you do not want random users assigned the role of ManagersGroup. Therefore, for this rule, you need to select *False*.

6 In the *Actions* section, click *Activate Role*.

7 In the *Activate Role* box, type `ManagersGroup`, then click *OK*.

The name you enter in the box is the role you want assigned to the users who match the condition.

Your rule should look similar to the following:

8 Click *OK* twice, then click *Apply Changes*.

9 To enable the role so that it can be used in Authorization and Identity Injection policies, click *Devices* > *Identity Servers* > *Servers* > *Edit* > *Roles*.

10 Select the check box next to the name of the role, then click *Enable*.

11 Click *OK*.

12 Update the Identity Server.

You can now use this role when creating Authorization and Identity Injection policies, which control access to protected Web resources. For more information, see:

- Chapter 3, "Creating Authorization Policies," on page 65
- Chapter 4, "Creating Identity Injection Policies," on page 113

2.5 Mapping Roles between Trusted Providers

The Identity Server can send roles in an authentication assertion. You can map these roles that are received from trusted providers to your own roles. Figure 2-14 illustrates this process.

Figure 2-14 *Role Mapping*

In this example, employees authenticate to identity providers novell.com (Liberty) or xyz.com (SAML 2.0). Each user is assigned to a role, such as N_EmployeeRole or XYZ_Empl. Attribute sets at each of the identity providers are configured to exchange the *All Roles* attribute with the trusted service provider, DigitalAirlines.com. DigitalAirlines.com consumes the authentication assertions, then maps the incoming roles to local roles. The mapped roles at DigitalAirlines.com can be used as evaluated conditions in authorization or J2EE policies, which can provide access to resources intended for the authenticated employees.

- Section 2.5.1, "Prerequisites," on page 61
- Section 2.5.2, "Procedure," on page 62

2.5.1 Prerequisites

❒ Configure trust between trusted providers, using the Liberty or SAML 2.0 protocol.

You should be familiar with "Configuring SAML and Liberty Trusted Providers" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

❒ Configure local authentication.

You must create an external contract at the service provider that matches the contract of the identity provider. See "Configuring Local Authentication" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

❒ Create an attribute set and select the local attribute *All Roles* in the set. This must be done at the identity provider and service provider.

This attribute set is used to pass roles from an identity provider to an external service provider in authentication assertions. See "Configuring Attribute Sets" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

2.5.2 Procedure

The following procedure describes how the service provider configures this type of role policy for novell.com, mapping the N_Employee role to an Access Manager role:

1. In the Administration Console, click *Policies* > *Policies*.
2. Click *New*, then specify a name for the Role policy.
3. Select *Identity Server: Roles* for the type, then click *OK*.
4. Configure the role policy as shown on the following page.

5. In the *Conditions* section, click *New* > *Roles from Identity Provider*.
6. Select the trusted identity provider in the drop-down menu.
7. For *Comparison*, select *String* > *Equals*.
8. Select *Value* > *Data Entry Field*.
9. Type the name of the role used by the trusted identity provider.
10. Under the *Actions* section, click *Activate Role*.
11. Type the name of the role you want to activate at the trusted service provider.
12. Click *OK*.
13. On the Policies page, click *Apply Changes*.
14. To enable the role so that it can be used in Authorization and Identity Injection policies, click *Identity Servers* > *Servers* > *Edit* > *Roles*.
15. Select the check box next to the name of the role, then click *Enable*.
16. Click *OK*.
17. Update the Identity Server.

2.6 Enabling and Disabling Role Policies

In order for a role policy to function, you must enable it for the Identity Server configuration.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Roles*.
2. Select the role policy's check box, then click *Enable*.
3. To disable the role policy, select the role policy's check box, then click *Disable*.
4. After enabling or disabling role policies, update the Identity Server configuration on the *Servers* tab.

2.7 Importing and Exporting Role Policies

You can import and export role policies in order to run them in other Identity Server configurations. When you import a role, ensure that you have enabled any Liberty profile that is referenced in the role policy, in order to correctly display the policy in the interface. However, the policy still evaluates if you have not enabled the profile.

You must also enable roles after importing them to an Identity Server configuration. See Section 2.6, "Enabling and Disabling Role Policies," on page 63. Click *Devices* > *Identity Servers* > *Edit* > *Roles*.

When you export a role policy, the system saves it as a `.txt` file at the location of your choosing. After you import a role policy, you must update the Identity Server configuration.

To export a role policy:

1. In the Administration Console, click *Policies* > *Policies*.
2. Select a policy, then click *Export*.
3. (Optional) Modify the name suggested for the file.
4. Click *OK*.
5. Using the features of your browser, specify where you want the file to be copied.
6. Click *OK*.

To import a role policy:

1. In the Administration Console, click *Policies* > *Policies*.
2. Click *Import*, then browse to and select the file.
3. Click *OK*.
4. When the policy appears in the list, click *Apply Changes*.

3 Creating Authorization Policies

Authorization policies are used when you want to protect a resource based on criteria other than authentication, and you want Access Manager to enforce the access restrictions. Authorization policies are enforced when a user requests data from a resource.

Access Manager supports three types of Authorization policies:

- Access Gateway Authorization policies for protecting resources of the Access Gateway
- Web Authorization policies for protecting Java applications on a J2EE server
- Enterprise JavaBean Authorization policies for protecting the Enterprise JavaBeans of a J2EE application

The first step in creating an Authorization policy is determining the criteria for restricting access. The second step is translating those criteria into rules and conditions for a policy. This section describes the policy elements, but your resource and your security requirements determine which elements to use when creating the policy.

- Section 3.1, "Designing an Authorization Policy," on page 65
- Section 3.2, "Creating Access Gateway Authorization Policies," on page 75
- Section 3.3, "Sample Access Gateway Authorization Policies," on page 77
- Section 3.4, "Creating Web Authorization Policies for J2EE Agents," on page 84
- Section 3.5, "Creating Enterprise JavaBean Authorization Policies for J2EE Agents," on page 86
- Section 3.6, "Conditions," on page 87
- Section 3.7, "Importing and Exporting Authorization Policies," on page 112

3.1 Designing an Authorization Policy

When you create an Authorization policy, you need to configure one or more rules. Each rule consists of two parts: (1) one or more conditions the user must meet and (2) the action to perform when the user meets the conditions or doesn't meet the conditions. The action can be to either allow or deny access to the resource. This section describes how to use the following elements when creating a policy:

- Section 3.1.1, "Controlling Access with a Deny Rule and a Negative Condition," on page 66
- Section 3.1.2, "Configuring the Result on Condition Error Option," on page 67
- Section 3.1.3, "Many Rules or Many Conditions," on page 67
- Section 3.1.4, "Using Multiple Conditions," on page 67
- Section 3.1.5, "Controlling Access with Multiple Conditions," on page 69
- Section 3.1.6, "Using Permit Rules with a Deny Rule," on page 70
- Section 3.1.7, "Using Deny Rules with a General Permit Rule," on page 71
- Section 3.1.8, "Public Policies," on page 73

- Section 3.1.9, “General Design Principles,” on page 73
- Section 3.1.10, “Using the Refresh Data Option,” on page 74
- Section 3.1.11, “Assigning Policies to Resources,” on page 74

3.1.1 Controlling Access with a Deny Rule and a Negative Condition

To deny access to the correct set of users, you need to know the characteristics of the users you don’t want to access the resource, as well as the characteristics of the users you do want to access the resource.

Some very simple policies can be created by using a Deny action. For example, suppose you have an application that you only want managers to access. If you have set up a role that assigns all managers to the Manager role, you can use this characteristic for an Authorization policy. Such a rule would be similar to the following:

Figure 3-1 *Simple Rule*

This rule evaluates the user, and if the user does not belong to the Manager role, the user matches the condition. The action for matching the condition is to deny access. The managers, who belong to the Manager role, do not match the condition and the Deny action is not applied to them.

The *Result on Condition Error* option is set to True. You don’t want an error to cause the policy to assume that the user is a manager. If an error occurs, you want the policy to assume that the user is not a manager, so he or she matches the condition and the Deny action is applied.

3.1.2 Configuring the Result on Condition Error Option

The *Result on Condition Error* option allows you to specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. You need to analyze the logic of your policy carefully, because if you set up this option incorrectly, error conditions can allow access to a resource. Consider the following:

- If your rule is a Permit rule and you do not want the action applied when an error occurs, select *False* for this option.
- If your rule is a Deny rule with an *If Not* condition and you want the action applied when an error occurs, select *True*.

3.1.3 Many Rules or Many Conditions

You can design your policy to have many rules with a single condition and action, or you can design your policy to have fewer rules, with each rule containing many conditions.

For example, suppose you have a resource that you don't want users accessing on Monday, Wednesday, and Friday between 1:00 a.m. and 2:00 a.m. You could set up three rules, one for each day, or you could set up one rule with three conditions. If all the conditions have the same action (for example, deny access with the same reason), it is simpler to put them in the same rule. However, if you have a customized message to return for each day, you need to put them in separate rules.

Each rule contains the following:

- Zero or more conditions. A condition specifies how the request data is evaluated for a True or False match. Conditions are evaluated in the order in which they are listed.
- One or more condition groups. Conditions are placed in condition groups, which gives you the flexibility of creating a policy that allows the user to match the conditions in one group but not the conditions in the other condition groups. Or you can set up the condition groups to require that the user matches at least one condition in each condition group.
- An action, which grants access, denies access, or redirects the users.

Conditions, conditions groups, and the interaction among them allow you to create very simple rules (if A, then grant access) to very complex rules (if A, B, and C, but not D and E, then grant access).

3.1.4 Using Multiple Conditions

The *Condition structure* option controls how conditions within a condition group interact with each other and how condition groups interact with each other. Select one of the following:

- **AND Conditions, OR groups:** If the conditions are ANDed, the user must meet all the conditions in a condition group to match the profile. If the condition groups are ORed, the user must meet all of the conditions of one group to match the profile. This option allows you to set up two or more profiles into which a user could fit and be considered a match. For example, suppose you create the following Permit rule:

  The first condition group contains the following conditions:

  1. The user's department must be Engineering.
  2. The request must come on a weekday.

The second condition group contains the following conditions:

1. The user's department must be Information Services and Technology (IS&T).
2. The request must come on a weekend.

With this rule, the engineers who match the first condition group have access to the resource during the week, and the IS&T users who match the second condition group have access to the resource on the weekend.

- **OR Conditions, AND groups:** If the conditions are ORed, the user must meet at least one condition in the condition group to match the profile. If the conditions groups are ANDed, the user must meet at least one condition in each condition group to match the profile. For example, suppose you create the following allow rule:

  The first condition group contains the following conditions:

  1. The user's department is Engineering.
  2. The user's department is Sales.

  The second condition group contains the following conditions:

  1. The user has been assigned the Party Planning role.
  2. The user has been assigned the Vice President role.

  With this rule, the Vice Presidents of both the Engineering and Sales departments can access the resource, and the users from the Engineering and Sales department who have been assigned to the Party Planning role can access the resource.

At the top of each condition group, there is an option that allows you to control whether the user must match the conditions to match the profile or whether the user matches the profile if the user doesn't match any of the conditions. Depending upon your selection for the Condition structure, you can select from the following:

- If/If Not
- Or/Or Not
- And/And Not

Conditions also have similar Not options, so that a user can match a condition by not matching the specified value.

Adding Multiple Conditions

To add another condition to a condition group, click *New*, then select a condition. To copy an existing condition, click the *Copy Condition* icon. New conditions are always added to the end of the condition group. Use the *Move* buttons to order the conditions in the condition group.

Adding New Condition Groups

To add another condition group to the rule, click *Append New Group*. To copy the existing condition group, click the *Copy Group* icon. New condition groups are always added to the end to the Conditions section. Use the *Move* buttons to order the condition groups.

Disabling or Moving Conditions and Condition Groups

Condition groups and conditions within them can be disabled by clicking the Enabled check mark, which changes the icon to the *Disabled* icon.

You usually disable a condition or condition group when testing a new rule, and if you decide that the condition or condition group is not needed, you can then use the *Delete* button to delete the condition or condition group from the rule. Use the *Move* buttons next to the *Delete* button to move a condition up or down within its group. Condition groups also have *Move* buttons.

3.1.5 Controlling Access with Multiple Conditions

A policy requires multiple conditions when you have more than one required condition for granting access. For example, suppose you can easily identity your managers because they have all been assigned the role of Manager, and you have a resource that only the sales managers should access. Such a policy requires two conditions for granting access: the Manager role and membership in the sales department. For a Deny rule, the rule needs two condition groups:

- The first condition group matches all users who are not managers. This causes the Deny action to be applied.
- The second condition group matches the users who are managers but don't belong to the sales department. Because they match both conditions, the Deny action is applied. For these two condition groups to work with this logic, the *Condition structure* is set to *AND Conditions, OR groups*.

The users who are managers and who belong to the sales department do not match either condition group. The Deny action is not applied, and they are allowed access.

Such a rule would look similar to the following:

Figure 3-2 *A Rule with Two Condition Groups*

This second condition group could be implemented as the second rule of the policy. If so, it should be set as a lower priority than the first rule. Because most systems would have more users than managers, the user rule would be used more frequently, so it should come first.

3.1.6 Using Permit Rules with a Deny Rule

You can also create policies that contain one or more Permit rules and then create the lowest priority rule in the policy as a Deny rule with no conditions. In such a policy, as soon as an allow match is processed, the rest of the rules are not processed and the user is granted access to the resource. The Deny rule is only processed if the user does not match one of the allow rules, and because all users match a rule with no conditions, the user is denied access to the resource.

The first rule in such a policy for the sales application would look similar to the following.

Figure 3-3 *Rule 1 Granting Access*

The conditions in Rule 1 are ANDed, which requires the user to match both conditions before they are granted access to the resource. The priority is set to 1, so this rule is the first rule that the Access Gateway processes. The J2EE authorization policies use the same logic.

The second rule would look similar to the following:

Figure 3-4 *Rule 2 Denying Access*

Because this rule has no conditions, any user who does not match the first rule does match this rule and is denied access. The priority of this rule is set lower than the Permit rule so that the Permit rule is processed first.

3.1.7 Using Deny Rules with a General Permit Rule

You can also create policies that contain one or more Deny rules and then create the lowest priority rule in the policy as a Permit rule with no conditions. In such a policy, as soon as a Deny rule matches a user, the rest of the rules are not processed and the user is denied access to the resource. The Permit rule is only processed if the user does not match one of the Deny rules. Because all users match a rule with no conditions, the user is allowed access to the resource.

The key to creating this type of policy is making sure all the Deny rules match the users you do not want accessing the resource and making sure that the *Result on Error Condition* option is set correctly.

For example, suppose one of the Deny rules uses an LDAP attribute for the condition and that the attribute is a hatSize attribute. Some of your users do not have a hatSize attribute, so when they access the resource, the comparison generates an error. If *Result on Error Condition* option is set to False, the action (Deny) is not applied, and the next rule in the policy is processed. If that rule is the general Permit rule, then they are allowed access to the resource because they experienced an error. To prevent this behavior, you need to set the *Result on Error Condition* option to True, so that the Deny action is applied. Your rule then denies access to everyone whose hatSize attribute matches the specified value and everyone who does not have the attribute.

The Deny rule for such a policy would look similar to the following:

Figure 3-5 *Deny Rule Configured for Error Conditions*

For most people, Deny rules are harder to write than Permit rules. You not only need to carefully configure the *Result on Condition Error* option, you must also carefully consider the consequences of the condition not matching a user. When a user doesn't match the condition, the Action is not applied and the next rule in the policy is evaluated. For example, suppose the URL condition is set to the compare the following value:

`http://sales.provo.novell.com/meetings/?`

If the URL in the request is `http://sales.provo.novell.com/meetings/january`, the user does not match the condition, because the ? applies only to the files in the `meetings` directory and not to the subdirectories. The Action is not applied, and the next rule or policy is evaluated. Consider the following possibilities:

- If you want the condition to match all files and subdirectories, you need to change the ? wildcard to the wildcard.
- If you want the condition to allow access to the files in the `/meetings` directory but deny access to the subdirectories, you need to negate the condition so it evaluates as follows: if the URL is not a request for the `/meetings/?` directory, deny access. If you select this type of condition, you need to set the *Result on Condition Error* option to True. If the comparison returns an error and there is the possibility that the request is for a subdirectory, you want the user to be denied access.

The general Permit rule for a Deny policy would look similar to the following:

Figure 3-6 *General Permit Rule*

NOTE: This type of policy is not recommended for WebSphere applications protected by the J2EE Agent. WebSphere, even when the user is logged in, always uses the anonymous user first to access resources, and switches to the actual username only when the anonymous user is denied. If the policy uses conditions that require information that is available only if the user is authenticated, this type of policy produces unexpected results.

3.1.8 Public Policies

You can create public authorization policies, which are policies that apply to everyone, by leaving the *Condition* section empty. In the *Action* section, you specify either to deny or to permit access to the resource. Then you assign the policy to the protected resource.

3.1.9 General Design Principles

When you design a policy, remember the following principles:

- Logged-in users are allowed access to a protected resource unless the policy denies access.
- Priority determines the order in which rules are applied.
- The Conditions section of the rule must evaluate to True in order for the Action section to be applied. If the Condition section evaluates to False, the Action section is ignored and the policy moves to the next rule. If another rule does not exist, the user is granted access to the resource.
- Rules are only processed until a user matches the conditions in a rule and its action is applied. If a user matches the first rule in a policy, that action is applied, and the rest of the rules in the policy are ignored.
- If two rules have the same priority, Deny rules are applied before Permit rules.
- After you have designed your policy, created it, and assigned it to a resource, you need to test the policy. You need to log in as the type of user who should be granted access, as the type of user who should not be granted access, and as a user who generates an error on condition evaluation.

3.1.10 Using the Refresh Data Option

Authorization policies are processed when a user requests access to a resource. The results and the values of the data items are cached for the user session. This means that when the user requests a second time to access the resource, the policy is evaluated, but the data values from the first evaluation are used. When a data item is cached for the user session, the user must log out and log back in to trigger new data values. (For information on how long the data items are cached, see Section 7.4, "The Policy Is Using Old User Data," on page 182.)

The LDAP Attribute can be configured to refresh its value according to a specified interval. This means the attribute value is read not just on the first request that triggers the policy evaluation, but when the interval expires. You can select to cache the value for the user session, the current request, or a time interval varying from 5 seconds to 60 minutes.

If the requested page contains links, you should usually cache the data for more than a single request. Each link on the page generates a new request.

You can use this feature for situations when you do not want to force the user to log in again to gain rights to resources or to revoke rights to resources. For example, suppose that you have an Authorization policy that grants access based on an LDAP attribute having a "yes" value. Users with a "no" value in this attribute are denied access.

If you don't enable the Refresh Data option on this attribute in the policy condition, the policy is evaluated when the user first tries to access the resource. The value for the attribute is cached for the user session, and until the user logs out, that is the value that is used.

However, if you enable the Refresh Data option on this attribute in the policy condition, the policy is evaluated when the user first tries to access the resource. When the user sends a second request to access the resource and the cached value has been marked old, the Refresh Data option causes the value of the attribute to be read again from the LDAP server. This new value is used to evaluate the policy and any other policy that is triggered by the request.

- If the value from the first request to the second request changes from no to yes, the user gets access to the resource.
- If the value from the first request to the second request changes from yes to no, the user is denied access to the resource.

For example:

- If the attribute controls access to employee resources and an employee leaves, a quick change of this attribute value cuts the employee off from the resources that should be available to employees only.
- If the attribute controls access to a software download site and a user has just purchased a product, a quick change to this attribute value can grant access to the download site.

IMPORTANT: This feature needs to be used with caution. Because querying the LDAP server slows down the processing of a policy, LDAP attribute values are normally cached for the user session. Enable this option only on those attributes that are critical to the security of your system or to the design of your work flow.

3.1.11 Assigning Policies to Resources

For information on how to assign thepolicy to a resource, see the following:

- For an Access Gateway policy, see "Assigning an Authorization Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.

- For a Web Authorization policy, see "Assigning a Web Authorization Policy to the Resource" in the *NetIQ Access Manager 3.2 J2EE Agent Guide*.
- For a Enterprise JavaBean Authorization policy, see "Assigning an Enterprise JavaBeans Authorization Policy to a Resource" in the *NetIQ Access Manager 3.2 J2EE Agent Guide*.

3.2 Creating Access Gateway Authorization Policies

An Authorization policy specifies conditions that a user must meet in order to access a resource or to be denied access to a resource. The Access Gateway enforces these conditions.

To create an Authorization policy:

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New*.
3. Specify a name for the policy, then select *Access Gateway: Authorization* for the type of policy.
4. Fill in the following fields:

   Description: (Optional) Describe the purpose of this rule.

   Priority: Specify the order in which a rule is applied in the policy, when the policy has multiple rules. The highest priority is 1 and the lowest priority is 10. If two rules have the same priority, a Deny rule is applied before a Permit rule.
5. In the *Condition Group 1* section, click *New*, then select one of the following:
   - **Authentication Contract:** Allows you to control access based on the contract the user used for login. For configuration information, see Section 3.6.1, "Authentication Contract Condition," on page 88.
   - **Client IP:** Allows you to control access based on the IP address of the client making the request. For configuration information, see Section 3.6.2, "Client IP Condition," on page 90.
   - **Credential Profile:** Allows you to control access based on the credentials the user specified during authentication. For configuration information, see Section 3.6.3, "Credential Profile Condition," on page 91.
   - **Current Date:** Allows you to control access based on the date of the request. For more information, see Section 3.6.4, "Current Date Condition," on page 92.
   - **Day of Week:** Allows you to control access based on the day the request is made. For configuration information, see Day of Week Condition.
   - **Current Day of Month:** Allows you to control access based on the month the request is made. For configuration information, see Section 3.6.6, "Current Day of Month Condition," on page 95.
   - **Current Time of Day:** Allows you to control access based on the time the request was made. For configuration information, see Section 3.6.7, "Current Time of Day Condition," on page 95.
   - **HTTP Request Method:** Allows you to control access based on the request method. For configuration information, see Section 3.6.8, "HTTP Request Method Condition," on page 97.
   - **LDAP Attribute:** Allows you to control access based on the value of an LDAP attribute. For configuration information, see Section 3.6.9, "LDAP Attribute Condition," on page 98.
   - **LDAP OU:** Allows you to control access based on the value of an LDAP organizational unit. For configuration information, see Section 3.6.10, "LDAP OU Condition," on page 99.

- **Liberty User Profile:** Allows you to control access based on the value of a Liberty attribute. For configuration information, see Section 3.6.11, "Liberty User Profile Condition," on page 100.
- **Roles:** Allows you to control access based on the roles a user has been assigned. For configuration information, see Section 3.6.12, "Roles Condition," on page 101.
- **URL:** Allows you to control access based on the URL in the request. For configuration information, see Section 3.6.13, "URL Condition," on page 102.
- **URL Scheme:** Allows you to control access based on the scheme in the URL of the request (for example, HTTP or HTTPS). For configuration information, see Section 3.6.14, "URL Scheme Condition," on page 103.
- **URL Host:** Allows you to control access based on the hostname in the URL of the request. For configuration information, see Section 3.6.15, "URL Host Condition," on page 105.
- **URL Path:** Allows you to control access based on the path in the URL of the request. For configuration information, see Section 3.6.16, "URL Path Condition," on page 106.
- **URL File Name:** Allows you to control access based on the filename in the URL of the request. For configuration information, see Section 3.6.17, "URL File Name Condition," on page 107.
- **URL File Extension:** Allows you to control access based on the file extension in the URL of the request. For configuration information, see Section 3.6.18, "URL File Extension Condition," on page 108.
- **X-Forwarded-For IP:** Allows you to control access based on the value in the X-Forwarded-For IP header of the HTTP request. For configuration information, see Section 3.6.19, "X-Forward-For IP Condition," on page 110.
- **Condition Extension:** (Conditional) If you have loaded and configured an authorization condition extension, this option specifies a condition that is evaluated by an outside source. This outside source returns either True or False. See the documentation that came with the extension for information about what is evaluated.
- **Data Extension:** (Conditional) If you have loaded and configured an authorization data extension, this option specifies the value that the extension retrieves. You can then select to compare this value with an LDAP attribute, a Liberty User Profile attribute, a Data Entry Field, or another Data Extension. For more information, see the documentation that came with the extension.

6 To add multiple conditions to the same rule, either add a condition to the same condition group or create a new condition group. For information on how conditions and condition groups interact with each other, see Section 3.1.4, "Using Multiple Conditions," on page 67.

7 In the *Actions* section, select one of the following:

- **Permit:** Allows the user to access the resource.
- **Redirect:** Specify the URL to which you want users redirected when they meet the conditions of this policy.
- **Deny:** Select one of the following deny actions:

  Display Default Deny Page: Displays a generic message, indicating that the user has insufficient rights to access the resource.

  Deny Message: Allows you to provide a customized message that is displayed to users who are denied access.

  Redirect to URL: Allows you to specify a URL that users are redirected to when they are denied access. For example:

  ```
  http://www.novell.com
  ```

- **Action Extension (Permit):** Select an action from the list of permit extensions. This action permits access to the resource and performs the additional action that the extension is designed to perform. If an action extension is not available, see Section 1.7, "Adding Policy Extensions," on page 16 for information on uploading, configuring, and importing extensions.
- **Action Extension (Deny):** Select an action from the list of deny extensions. This action denies access to the resource and performs the additional action that the extension is designed to perform. If a deny extension is not available, see Section 1.7, "Adding Policy Extensions," on page 16 for information on uploading, configuring, and importing extensions.

8 (Conditional) If you have installed an action obligation extension, you can click *New* in the *Actions* section, and select the action. This causes the extension to perform whatever action it is designed to perform whenever a user matches the conditions of this rule. This type of action is usually always configured in addition to a permit or deny action. If the obligation option is not available, see Section 1.7, "Adding Policy Extensions," on page 16 for information on uploading, configuring, and importing extensions.

9 To save the rule, click *OK*.

10 To add another rule, click *New* or to save the policy, click *OK*, then click *Apply Changes*.

11 Assign the policy to a protected resource (see "Assigning an Authorization Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide*).

3.3 Sample Access Gateway Authorization Policies

- Section 3.3.1, "Sample Policy Based on Organizational Rules," on page 77
- Section 3.3.2, "Sample Workflow Policy," on page 81

3.3.1 Sample Policy Based on Organizational Rules

The following sections describe a scenario with an organizational division, then describe two types of policies that enforce the requirements of the scenario:

- "Company Scenario" on page 77
- "LDAP Context Policies" on page 78
- "Role Policies with Authorization Policies" on page 79

Company Scenario

Suppose that the company LDAP directory has the following organization:

ou=sales,o=acme
ou=dev,o=acme
ou=hr,o=acme

Suppose that this company has the following configuration and requirements:

- Under each branch of the tree, the system administrator has created the users who work in these departments.
- Each department has its own Web resources, and other departments must be denied access to these resources.

With this type of configuration, you can use the LDAP context condition to create authorization policies or you can create role policies that are used in conjunction with authorization policies.

LDAP Context Policies

With such an organization, you can create a policy that either allows or denies access based on the LDAP context of the user's DN. You can use the LDAP context of the user DN to separate the users into their departments and then grant access based on the context match. You need to create protected resources for the Web resources of the department, create a policy for each protected resource, and assign a policy to the protected resources.

The following procedure explains how to configure such a policy for the sales department.

1. Click *Policies* > *Policies* > *New*, specify a name for the policy, select *Access Gateway: Authorization* as the type, then click *OK*.
2. For *Condition Group 1*, click *New*, then select *Credential Profile*.
3. Fill in the following fields:

   LDAP Credentials: Select *LDAP User DN*.

   If/If Not: Select *If Not*.

   Comparison: Select *Contains Substring*.

   Mode: Select *Case Insensitive.*

   Value: Select *Data Entry Field*. In the text box, type the following value:

   ```
   ou=sales,o=acme
   ```

   Result on Condition Error: Select *True*.
4. In the *Actions* section, select *Deny*.

Your policy should look similar to the following:

This sets up the condition so that the following occurs:

- When the user does not belong to the sales department, the user is denied access.
- When the user belongs to the sales department, the user is granted access.
- When an error occurs evaluating the conditions in the rule, the user is denied access.

5 Assign the policy to the protected Web resources of the sales department (see "Assigning an Authorization Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide*).

6 Repeat these steps for the other two departments, changing the *Value* field to match the appropriate department.

Role Policies with Authorization Policies

Because of the company's organization, you need to create three role policies, one for the sales users, one for the development users, and one for the human resource users. You can then use these roles as conditions in authorization policies to allow and deny access. The first time you use roles in an authorization policy, there is extra setup because you must create the role policies. However, after the role policies are created, you can use them in multiple authorization policies.

The following instructions explain how to use the Sales role to create a policy that controls access to a protected resource. For instructions on how to create the Sales role, see "Creating a Role by Using the Location of the User Objects" on page 56.

You need to decide on the type of Authorization policy you want to create. For example, you can create a Deny policy that denies access to everyone who does not match the condition (in this case, the Sales role).

Or you can create a two-rule policy that allows access to everyone that matches the condition. The first rule grants access to everyone who has the Sales role, and the second rule denies access to everyone who did not match the conditions of the first rule. (Other methods are also possible.) Because the proposed Deny policy is very similar to the LDAP Context Policies example, the following procedures explain how to create the two-rule policy.

1. In the Administration Console, click *Policies* > *Policies* > *New*.
2. Specify a name for the policy, select *Access Gateway: Authorization* as the type, then click *OK*.
3. (Optional) Provide a description for the rule.
4. In *Condition Group 1*, click *New*, and select *Roles*.
5. Fill in the following fields:

   If/If Not: Select *If*.

   Roles: Select *[Current]*.

   Comparison: Select *String: Equals*.

   Mode: Select *Case Insensitive*

   Value: Select *Roles,* then select *Sales.*

   Result on Condition Error: Select *False*.
6. Under *Actions*, select *Permit*, then click OK.

   These steps create the Permit rule and set up the condition so that the following occurs:

   - When the user does not match the condition because the user does not belong to the Sales role, the policy engine moves to the next rule in the policy.
   - When the user does match the condition because the user belongs to the Sales role, the user is granted access.
   - If an error occurs when evaluating the condition of the policy, the user does not match the condition and the policy engine moves to the next rule in the policy.
7. In the *Rule List*, click *New*.

   This second rule is for denying access to everyone who does not match the condition in Rule 1. Processing of the policy stops when a user matches a rule; therefore all users who match Rule 1 are granted access and the policy engine does not evaluate the second rule.
8. Set the *Priority* to be 2 or greater.

   You want the Permit rule to be processed first, so it should have a priority of 1. The Deny rule needs to be processed last, so it needs a lower priority than the Permit rule.
9. Leave the *Condition Group 1* empty.

   The *Conditions* section is left empty so that everyone who does not match the conditions of the Permit rule is denied access to the resource.
10. In the *Actions* section, select *Deny* and either accept the default action or select one of the other actions.
11. Click *OK* twice.
12. Click *Apply Changes* on the Policies page.
13. Assign the policy to the protected Web resources of the sales department (see "Assigning an Authorization Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide*).

3.3.2 Sample Workflow Policy

One of the common workflow problems that an Authorization policy can solve is what to do with users who are denied access to resource. Most of the time they have a legitimate reason for trying to access the resource and need contact information to request access to the resource. You can add this contact information to a Web page and redirect the users to this page when the policy denies the user access.

To create such a workflow, you need to create an HTML page with the necessary information for making the request for access. It can be as simple as a contact name or it can be an actual form that the user submits to the organization that controls access to the resource.

You then need to create an Authorization policy that redirects the denied users to this page. The following sample policy uses a role for the access condition, but the same workflow can be created using any of the other conditions available for an Authorization policy. For this example, assume that the user is granted a Master role if the user is a member of the Master group. The organization that controls access to the resource is the owner of the Master group and can add and delete members from the group. When the owner of the Master group receives a request for access to the resource, the owner can evaluate the user, and if the user meets their standards, the owner adds the user to the Master group.

You can use the Master group to create an Access Manager Role policy. This policy for the Master role should look similar to the following:

Figure 3-7 *A Role Policy with an LDAP Group Condition*

This rule grants the user the Master role if the user belongs to the cn=Master,o=novell LDAP group. If the user doesn't belong to this group or if an error occurs trying to get the data, the user is not assigned the role. This occurs because both the condition and the *Result on Condition Error* evaluate to False, which prevents the Action from being applied.

After creating the Role policy, apply the changes and enable the Role for the Identity Server.

You can then use this role to create an Authorization policy that contains two rules. The first rule grants access to the users who have the Master role (and are therefore members of the Master group). This rule should look similar to the following:

Figure 3-8 *A Permit Rule with a Role Condition*

This rule permits users who are assigned the Master role to have access to the resource. If the user does not match the condition or if an error occurs accessing the user's role information, the user is sent to the next rule because both the condition and the *Result on Condition Error* evaluate to False.

The second rule in the policy should deny access to those who are not assigned the Master role and should redirect them to the page where they can request access. You can do this with a rule that checks to see if they are assigned the Master role. In this type of rule, the condition needs to be an *If Not* condition.

Figure 3-9 *A Deny Rule with a Redirect URL*

With an *If Not* condition, the condition evaluates to True when the user does not match the condition. With such a rule, you want the *Result on Condition Error* to also evaluate to True. If there is an error obtaining role information for the user, you don't want the rule to assume that the user had the Master role. You want the rule to assume that the user had no roles, or in other words, you want the error condition to evaluate to True.

Because the condition evaluated to True, the Action is applied to the user. The value specified in the *Redirect to URL* text box should specify the page that contains the information on how to request access.

This redirect rule could be the only rule in the Authorization policy, because the users who are assigned to the Master role do not match the rule and are thus allowed access. Having the first rule that grants access because they have the Master role just makes the logic of the policy clearer.

If you create the first rule that grants users with the Master role access, you can use a general Deny rule for the second rule.

It should look similar to the following:

Figure 3-10 *A General Deny Rule*

A general Deny rule has no conditions, so it matches everyone that does not match the first rule in the policy. You can add more rules to this policy to tighten security so that not all users are redirected to the site that contains the information on how to request access. For this type of policy, the last rule would be a general Deny rule with no conditions and without a redirect. The rules between Rule 1, which granted access to people assigned to the Master role, and the last rule, which denies everyone, should be rules that identify the types of users who have legitimate reasons for requesting access, and these rules should contain the redirect action.

After you have saved the Authorization policy, you need to assign it to the protected resource or resources that require the Master role, then update the Access Gateway.

3.4 Creating Web Authorization Policies for J2EE Agents

A Web Authorization policy specifies the criteria a user must meet to either allow access or deny access to a resource. For example, if you create a Sales role and assign it to the users, the role can be used to allow access to the sales applications and to deny access to resource management applications.

To create a Web Authorization policy:

1. In the Administration Console, click *Policies* > *Policies* > *New*.
2. Specify a name for the policy, select *J2EE Agent: Web Authorization* as the type, then click *OK*.
3. Fill in the following fields:

   Description: (Optional) Specify a description for the rule.

   Priority: Specify the order in which a rule is applied in the policy, when the policy has multiple rules. The highest priority is 1 and the lowest priority is 10. If two rules have the same priority, a Deny rule is applied before a Permit rule.

4 In the *Condition Group 1* section, click *New*, then select one of the following:

- **Client IP Address:** Allows you to control access based on the IP address of the client making the request. For configuration information, see Section 3.6.2, "Client IP Condition," on page 90.
- **Credential Profile:** Allows you to control access based on the credentials the user specified during authentication. For configuration information, see Section 3.6.3, "Credential Profile Condition," on page 91.
- **Current Date:** Allows you to control access based on the date of the request. For more information, see Section 3.6.4, "Current Date Condition," on page 92.
- **Day of Week:** Allows you to control access based on the day the request is made. For configuration information, see Section 3.6.5, "Day of Week Condition," on page 93.
- **Current Day of Month:** Allows you to control access based on the month the request is made. For configuration information, see Section 3.6.6, "Current Day of Month Condition," on page 95.
- **Current Time of Day:** Allows you to control access based on the time the request was made. For configuration information, see Section 3.6.7, "Current Time of Day Condition," on page 95.
- **HTTP Request Method:** Allows you to control access based on the request method. For configuration information, see Section 3.6.8, "HTTP Request Method Condition," on page 97.
- **LDAP Attribute:** Allows you to control access based on the value of an LDAP attribute. For configuration information, see Section 3.6.9, "LDAP Attribute Condition," on page 98.
- **Liberty User Profile:** Allows you to control access based on the value of a Liberty attribute. For configuration information, see Section 3.6.11, "Liberty User Profile Condition," on page 100.
- **Roles:** Allows you to control access based on the roles a user has been assigned. For configuration information, see Section 3.6.12, "Roles Condition," on page 101.
- **URL:** Allows you to control access based on the URL in the request. For configuration information, see Section 3.6.13, "URL Condition," on page 102.
- **URL Scheme:** Allows you to control access based on the scheme in the URL of the request (for example, HTTP or HTTPS). For configuration information, see Section 3.6.14, "URL Scheme Condition," on page 103.
- **URL Host:** Allows you to control access based on the hostname in the URL of the request. For configuration information, see Section 3.6.15, "URL Host Condition," on page 105.
- **URL Path:** Allows you to control access based on the path in the URL of the request. For configuration information, see Section 3.6.16, "URL Path Condition," on page 106.
- **URL File Name:** Allows you to control access based on the filename in the URL of the request. For configuration information, see Section 3.6.17, "URL File Name Condition," on page 107.
- **URL File Extension:** Allows you to control access based on the file extension in the URL of the request. For configuration information, see Section 3.6.18, "URL File Extension Condition," on page 108.
- **X-Forwarded-For IP:** Allows you to control access based on the value in the X-Forwarded-For IP header of the HTTP request. For configuration information, see Section 3.6.19, "X-Forward-For IP Condition," on page 110.

5 To add multiple conditions to the same rule, either add a condition to the same condition group or create a new condition group.

6 In the *Actions* section, select either *Permit* or *Deny*.

7 To save the rule, click *OK* twice, then click *Apply Changes*.

8 Assign the policy to a Web resource. See "Assigning a Web Authorization Policy to the Resource" in the *NetIQ Access Manager 3.2 J2EE Agent Guide*.

3.5 Creating Enterprise JavaBean Authorization Policies for J2EE Agents

An Enterprise JavaBean (EJB) Authorization policy allows you to protect the entire bean or specific interfaces or methods.

To create an EJB Authorization policy:

1 In the Administration Console, click *Policies* > *Policies* > *New*.

2 Specify a name for the policy, select *J2EE Agent: EJB Authorization* as the type, then click *OK*.

3 Fill in the following fields:

Description: (Optional) Specify a description for the rule.

Priority: Specify the order in which a rule is applied in the policy, when the policy has multiple rules. The highest priority is 1 and the lowest priority is 10. If two rules have the same priority, a Deny rule is applied before a Permit rule.

4 In the *Condition Group 1* section, click *New*, then select one of the following:

- **Credential Profile:** Allows you to control access based on the credentials the user specified during authentication. For configuration information, see Section 3.6.3, "Credential Profile Condition," on page 91.
- **Current Date:** Allows you to control access based on the date of the request. For more information, see Section 3.6.4, "Current Date Condition," on page 92.
- **Day of Week:** Allows you to control access based on the day the request is made. For configuration information, see Section 3.6.5, "Day of Week Condition," on page 93.
- **Current Day of Month:** Allows you to control access based on the month the request is made. For configuration information, see Section 3.6.6, "Current Day of Month Condition," on page 95.
- **Current Time of Day:** Allows you to control access based on the time the request was made. For configuration information, see Section 3.6.7, "Current Time of Day Condition," on page 95.
- **LDAP Attribute:** Allows you to control access based on the value of an LDAP attribute. For configuration information, see Section 3.6.9, "LDAP Attribute Condition," on page 98.
- **Liberty User Profile:** Allows you to control access based on the value of a Liberty attribute. For configuration information, see Section 3.6.11, "Liberty User Profile Condition," on page 100.
- **Roles:** Allows you to control access based on the roles a user has been assigned. For configuration information, see Section 3.6.12, "Roles Condition," on page 101.

5 To add multiple conditions to the same rule, either add a condition to the same condition group or create a new condition group.

6 In the *Actions* section, select either *Permit* or *Deny*.

7 To save the rule, click *OK*, then click *Apply Changes*.

8 Assign the policy to an EJB resource. See "Assigning an Enterprise JavaBeans Authorization Policy to a Resource" in the *NetIQ Access Manager 3.2 J2EE Agent Guide*.

3.6 Conditions

This section describes the possible conditions for an Authorization policy. Some conditions can be set up so that the current values in the request are compared against static values (A to B), or you can compare static values to current values in the request (B to A). Within one policy, you should probably decide which direction to set up the comparisons and remain consistent unless there is a compelling reason to switch the direction for a particular condition.

For example, suppose you set up a rule to allow access to a resource only during the weekdays (Monday through Friday). You set up four of these conditions to compare if the date when the request is made matches with Monday, Tuesday, Wednesday, or Thursday. You set up the fifth condition to compare whether Friday matches the date when the request is made. This works, but maintaining this policy is more difficult because each new policy manager will look at the Friday condition and wonder why it is configured differently.

Many conditions, when used as the sole condition of a rule, do not make very useful rules. For example, you can create a rule that grants access if the user specifies a specific URL in the request. Such a rule has limited application. But a rule that requires that the request contain a specific URL and that the user have a specific role has greater application because it can be used to limit access to the URL based on the user's role. For information about how conditions can be ANDed or ORed together or placed in different condition groups, see Section 3.1.4, "Using Multiple Conditions," on page 67.

Authorization policies use the following conditions:

- Section 3.6.1, "Authentication Contract Condition," on page 88
- Section 3.6.2, "Client IP Condition," on page 90
- Section 3.6.3, "Credential Profile Condition," on page 91
- Section 3.6.4, "Current Date Condition," on page 92
- Section 3.6.5, "Day of Week Condition," on page 93
- Section 3.6.6, "Current Day of Month Condition," on page 95
- Section 3.6.7, "Current Time of Day Condition," on page 95
- Section 3.6.8, "HTTP Request Method Condition," on page 97
- Section 3.6.9, "LDAP Attribute Condition," on page 98
- Section 3.6.10, "LDAP OU Condition," on page 99
- Section 3.6.11, "Liberty User Profile Condition," on page 100
- Section 3.6.12, "Roles Condition," on page 101
- Section 3.6.13, "URL Condition," on page 102
- Section 3.6.14, "URL Scheme Condition," on page 103
- Section 3.6.15, "URL Host Condition," on page 105
- Section 3.6.16, "URL Path Condition," on page 106
- Section 3.6.17, "URL File Name Condition," on page 107
- Section 3.6.18, "URL File Extension Condition," on page 108
- Section 3.6.19, "X-Forward-For IP Condition," on page 110
- Section 3.6.20, "Condition Extension," on page 111
- Section 3.6.21, "Data Extension," on page 111

For the specific policies they can be used in, see the following:

- Section 3.2, “Creating Access Gateway Authorization Policies,” on page 75
- Section 3.4, “Creating Web Authorization Policies for J2EE Agents,” on page 84
- Section 3.5, “Creating Enterprise JavaBean Authorization Policies for J2EE Agents,” on page 86

3.6.1 Authentication Contract Condition

The Authentication Contract condition matches the contract the user logged in with to the contract specified in this condition. The Identity Server has the following default contracts:

Name	URI
Name/Password - Basic	basic/name/password/uri
Name/Password - Form	name/password/uri
Secure Name/Password - Basic	secure/basic/name/password/uri
Secure Name/Password - Form	secure/name/password/uri

To configure other contracts for your system, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts*.

To specify an Authentication Contract condition, fill in the following fields:

Authentication Contract: To compare the contract that the user used with a static value, select *Current*. To compare a static value with what the user used, select a contract from the list.

If you have created more than one Identity Server configuration, select the configuration that corresponds to the configuration your Access Gateway is configured to trust, then select the contract. The name of the contract is displayed. When you select this name, the configurations that contain a definition for this contract are highlighted.

If you select a contract that is defined on only one of your configurations, be aware that you must change this policy when you change configurations. If you select a contract that is defined in all your configurations, this policy requires no modifications and continues to function when you change configurations.

For example, the following policy has selected Name/Password - Basic as the contract:

Figure 3-11 *An Authentication Contract Defined by Multiple Identity Server Configurations*

Two Identity Server configurations have been defined (idp-43.amlab.net and idp-51.amlab.net). Both configurations are highlighted because Name/Password - Basic is a contract that is automatically defined for all Identity Server configurations. Because it is defined on both configurations, this policy's function is the same, regardless of which configuration is selected as the trusted configuration.

Comparison: Specify how the contract is compared to the data in the *Value* field. Select either a string comparison or a regular expression:

- **Comparison: String:** Specifies that you want the values compared as strings and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the Authentication Contract value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the Authentication Contract value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the Authentication Contract value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value you want to compare with the Authentication Contract value. If you select a static value for the Authentication Contract value, select *Authentication Contract* and *Current*. If you select *Current* for the Authentication Contract value, select *Authentication Contract*, then select the name of a contract.

Other value types are possible if you selected *Current* for the Authentication Contract value. For example:

- You can select *Data Entry Field*. The value specified in the text box must be the URI of the contract for the conditions to match. For a list of these values, click *Access Manager* > *Identity Servers* > *Edit* > *Local* > *Contracts*.
- If you have defined a Liberty User Profile attribute for the URI of authentication contracts, you can select *Liberty User Profile* and your defined attribute.
- If you have defined an LDAP attribute for the URI of the authentication contracts, you can select *LDAP Attribute* and your defined attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.2 Client IP Condition

The Client IP condition allows you to use the IP address of the user making the request to determine whether the user is allowed access to a resource.

Fill in the following fields:

Comparison: Specify how the client IP address is compared to the data in the *Value* field. Select either an IP comparison or a regular expression:

- **Comparison: IP:** Specifies that you want the values compared as IP addresses. Select one of the following:
  - **Equals:** Allows you to specify an IP address that the client must match. You can specify more than one.
  - **In Range:** Allows you to specify a range of IP addresses that the client's address must fall within. You can specify more than one range.
  - **In Subnet:** Allows you to specify the subnet that the client's address must belong to. You can specify more than one subnet.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions. If you select this option, you must also specify a mode. Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Select *Data Entry Field* and specify a value appropriate for your comparison type. Use the *Edit* button to access a text box where you can enter multiple values, each on a separate line. (For more information, see Section 3.6.23, "Edit Button," on page 112.) Use the *Add* button to add values one at a time. For example:

Comparison Type	Value
Equals	`10.10.10.10` `10.10.10.11`
In Range	`10.10.10.10 - 10.10.10.100` `10.10.20.10 - 10.10.20.100`
In Subnet	`10.10.10.12 / 22` `10.10.20.30 / 22`

Other values types are possible. For example, if your user store contains an LDAP attribute with the IP address of your users, you could select to compare the client's current IP address with the stored value by using an LDAP attribute or a Liberty User Profile value.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.3 Credential Profile Condition

The Credential Profile condition allows you to control access based on the credentials the user entered when authenticating to the system.

To set up the matching for this condition, fill in the following fields:

Credential Profile: Specify the type of credential your users are using for authentication. If you have created a custom contract that uses credentials other than the ones listed below, do not use the Credential Profile as a condition.

To configure the Credential Profile condition, select one of the following:

- **LDAP Credentials:** If you prompt the user for a username, select this option, then select *LDAP User Name* (the cn of the user), *LDAP User DN* (the fully distinguished name of the user), or *LDAP Password*.

  The default contracts assign the cn attribute to the Credential Profile. If your user store is an Active Directory server, the SAMAccountName attribute is used for the username and stored in the cn field of the LDAP Credential Profile.
- **X509 Credentials:** If you prompt the user for a certificate, select this option, then select one of the following:
  - **X509 Public Certificate Subject:** Retrieves the subject field from the certificate, which can match the DN of the user, depending upon who issued the certificate.
  - **X509 Public Certificate Issuer:** Retrieves the issuer field from the certificate, which is the name of the certificate authority (CA) that issued the certificate.
  - **X509 Public Certificate:** Retrieves the entire certificate, Base64 encoded.
  - **X509 Serial Number:** Retrieves the serial number of the certificate.
- **SAML Credential:** If your users authenticate using a SAML assertion, select this option.

Comparison: Select one of the following types:

- **Comparison: String:** Specifies that you want the values compared as strings and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the Credential Profile value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the Credential Profile value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the Credential Profile value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the second value for the comparison. Select one of the following data types:

- **LDAP Attribute:** If you have an LDAP attribute that corresponds to the Credential Profile you have specified, select this option and the attribute.
- **Liberty User Profile:** If you have a Liberty User Profile attribute that corresponds to the Credential Profile you have specified, select this option and the attribute.
- **Data Entry Field:** Specify the string you want matched. Be aware of the following requirements:
  - If you selected *LDAP User DN* as the credential, you need to specify the DN of the user in the *Value* text box. If the comparison type is set to *Contains Substring*, you can match a group of users by specifying a common object that is part of their DNs, for example ou=sales.
  - If you selected *X509 Public Certificate Subject* as the credential, you need to specify all elements of the Subject Name of the certificate in the *Value* text box. Separate the elements with a comma and a space, for example, o=novell, ou=sales. If the comparison type is set to *Contains Substring*, you can match a group of certificates by specifying a name that is part of their Subject Name, for example ou=sales.

Other values are possible. Your policy requirements determine whether they are useful.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.4 Current Date Condition

The Current Date condition allows you to use the date to determine whether the user is allowed access to a resource.

Fill in the following fields:

Comparison: Specify how the current date is compared to the data in the *Value* field. Select one of the following types:

- **Comparison: Date:** Specifies that you want the values compared as dates. Select one of the following date operators:
  - **Equals:** Requires that the current date must equal the specified value.
  - **Greater Than:** Requires that the current date be after the specified value.

- **Greater Than or Equal to:** Requires that the current date be after or equal to the specified value.
- **Less Than:** Requires that the current date be before the specified value.
- **Less Than or Equal to:** Requires that the current date be before or equal to the specified value.

- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions. Be aware the regular expression matching uses the entire date of the server in its matching. Therefore if the value you are matching is 8, the 8 can produce a match for the year (2008), the month (8), and the day (8, 18, 28).

  If you select this option, you must also specify a mode. Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Date Format: If you selected a date comparison, specify the format of the *Value* field. Select one of the following formats:

- **D/M/Y =** 1/Jul/2009 or 1/7/2009
- **D-M-Y =** 1-Jul-2009 or 1-7-2009
- **D.M.Y =** 1.Jul.2009 or 1.7.2009
- **M/D/Y =** Jul/1/2009 or 7/1/2009
- **M-D-Y =** Jul-1-2009 or 7-1-2009
- **M.D.Y =** Jul.1.2009 or 7.1.2009
- **YYYY-MM-DD =** 2009-07-01
- **YYYY.MM.DD =** 2009.07.01

D specifies a number from 1 to 31. *M* specifies a number from 1 to 12 or the name of the month in three letters (Sep) or complete (September). *Y* specifies the year in a four-digit format.

Value: Specify the second value for the comparison. If you select *Data Entry Field* as the value type, specify the date in the format you select in the *Date Format* field.

Other value types are possible. Your policy requirements determine whether they are useful. If you have set up a Liberty User Profile or an LDAP attribute that corresponds to the date, you can use this option and select your attribute. The *Date Format* field does not apply to these value types.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.5 Day of Week Condition

The Current Day of Week condition allows you to restrict access based on which day of the week the request is made. Fill in the following fields:

Current Day of Week: Select the name of the day from the list. To compare the day specified in the current request with a static value, select *Current*. To compare a static value with the day specified in the current request, select the name of a day from the list.

Comparison: Specify how the current day of the week is compared to the data in the *Value* field. Select one of the following types:

- **Comparison: Day of Week:** Specifies that you want the values compared as a day of the week. Select one of the following operators:
  - **Equals:** Allows you to specify a day that the client must match.
  - **In Range:** Allows you to specify a range of days that the client's request must fall within, for example, Monday to Friday.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions. Be aware that regular expression matching uses the entire date of the server in its matching. Therefore if the value you are matching is M, the M can produce a match for months (March and May) and for time zones (such as MST).

  If you select this option, you must also specify a mode. Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the second value for the comparison. If you select *Current* for the *Current Day of Week* field, you need to specify a static value. If you select a static value for the *Current Day of the Week* field, you need to select *Current* for the *Value* field.

If you select *Data Entry Field* as the value type, days of the week are specified in the following format:

```
Sun or Sunday
Mon or Monday
Tue or Tuesday
Wed or Wednesday
Thu or Thursday
Fri or Friday
Sat or Saturday
```

If you selected *In Range* as the comparison type, specify the first day of the range in the left text box and the end day of the range in the right text box.

Other value types are possible. Your policy requirements determine whether they are useful. If you have set up a Liberty User Profile or an LDAP attribute that corresponds to a day of the week, you can use this option and select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.6 Current Day of Month Condition

The Current Day of Month condition allows you to restrict access based on the day of the month the request is made. Fill in the following fields:

Comparison: Specify how the current day of the month is compared to the data in the *Value* field. Select one of the following types:

- **Comparison: Day of Month:** Specifies that you want the values compared as a day of the month. Select one of the following operators:
  - **Equals:** Allows you to specify a day that the client must match.
  - **In Range:** Allows you to specify a range of days that the client's request must fall within.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions. Regular expression matching uses the entire date of the server in its matching. Therefore if the value you are matching is 8, the 8 can produce a match for the year (2008), the month (8), and the day (8, 18, 28). If you want to match only on a day of the month (1-31), you need to use the Day of Month comparison rather than a Regular Expression comparison.

  If you select this option, you must also specify a mode. Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the second value for the comparison:

- If you select *Equals* for the comparison type, you would normally select *Data Entry Field* for the *Value* field and specify a number from 1 to 31 in the text box.
- If you select *In Range* for the comparison type, you would normally select *Data Entry Field* for the *Value* field and specify the first value of the range in the first text box and the second value of the range in the second text box. If you specify 1 in the first box and 15 in the second box, you can use this condition to restrict access between the first day of the month and the 15th day.

Other value types are possible. Your policy requirements determine whether they are useful. If you have set up a Liberty User Profile or an LDAP attribute that corresponds to a day of the month, you can use this option and select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.7 Current Time of Day Condition

The Current Time of Day condition allows you to restrict access based on the time the request is made. Fill in the following fields:

Comparison: Specify how the current time of day is compared to the data in the *Value* field. Select one of the following types:

- **Comparison: Time:** Specifies that you want the values compared as time. Select one of the following:
  - **Greater Than:** Requires that the current time is greater than the specified value.
  - **Greater Than or Equal to:** Requires that the current time is greater than or equal to the specified value.
  - **Less Than:** Requires that the current time is less than the specified value.
  - **Less Than or Equal to:** Requires that the current time is less than or equal to the specified value.
  - **In Range:** Requires that the current time must fall within the specified range, such as 08:00 and 17:00.

  If you specify this type of comparison, you must also specify a time zone. Select either the *Local* time zone or *GMT* (Greenwich Mean Time).
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions. Regular expression matching uses the entire date and time of the server in its matching. Therefore if the value you are matching is 8, the 8 can produce a match for the year (2008), the month (8), the day (8, 18, 28), the hour (8), the minute (8, 18, 28, 38, 48) and the second (8, 18, 28, 38, 48).

  If you select this option, you must also specify a mode. Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the second value for the comparison. If you select *Data Entry Field* as the value type, hours and minutes are specified in the following format:

```
hour:minute
```

Hour is a number from 00 to 23, and minute is a number from 00 to 59.

Time can only be specified in a 24-hour clock format. For example, 8 am is 08:00 and 5:30 pm is 17:30.

Other value types are possible. Your policy requirements determine whether they are useful. If you have set up a Liberty User Profile or an LDAP attribute that corresponds to the time of day, you can use this option and select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.8 HTTP Request Method Condition

The HTTP Request Method condition allows you to restrict accessed based on the request method in the current request.

HTTP Request Method: Select the request method from the list or select *Current* to specify the method in the current request.

Comparison: Specify how the HTTP Request Method is compared to the data in the *Value* field. Select one of the following types:

- **Comparison: String:** Specifies that you want the values compared as strings and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the HTTP Request Method value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the HTTP Request Method value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the HTTP Request Method value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value you want compared to the HTTP Request Method value. If you selected a method from the list for the HTTP Request Method value, select *HTTP Request Method* > *Current*. If you selected *Current* for the HTTP Request Method value, select a request method from the list.

Other value types are possible. Your policy requirements determine whether they are useful. If you have set up a Liberty User Profile or an LDAP attribute that corresponds to an HTTP Request Method, you can use this option and select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.9 LDAP Attribute Condition

The LDAP Attribute condition allows you to restrict access based on a value in an LDAP attribute defined for the inetOrgPerson class or any other LDAP attribute you have added. You can have the user's attribute value retrieved from your LDAP directory and compared to a value of the following type:

- Roles from an identity provider
- Date and time and its various elements
- URL and its various elements
- IP address
- Authentication contract
- Credential profile
- HTTP request method
- Liberty User Profile attribute
- Static value in a data entry field

This condition is one of the slower conditions to process because the value needs to be retrieved from the LDAP server. If the value is not time sensitive, you can have attribute value sent in the assertion when the user authenticates. Its value is then in cache and available. For configuration information, click *Devices > Identity Servers > Servers > Edit > Liberty [or SAML 1.0 or SAML 2.0] > [Provider] > Attributes*.

To set up the matching for this condition, fill in the following fields:

LDAP Attribute: Specify the LDAP attribute you want to use in the comparison. Select from the listed LDAP attributes. To add an attribute that isn't in the list, scroll to the bottom of the list, click *New LDAP Attribute*, then specify the name of the attribute.

Refresh Data Every: Sends a query to the LDAP server to verify the current value of the attribute according to the specified interval. Because querying the LDAP server slows down the processing of a policy, LDAP attribute values are normally cached after the value has been obtained. The default cache interval is for the user session. You should change the value of this option from Session to a more frequent interval only on those attributes that are critical to the security of your system or to the design of your work flow.

You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes. For more information on this option, see Section 3.1.10, "Using the Refresh Data Option," on page 74.

Comparison: Specify how you want the values compared. All data types are available. Select one that matches the value type of your attribute.

Mode: Select the mode, if available, that matches the comparison type. For example, if you select to compare the values as strings, you can select either a *Case Sensitive* mode or a *Case Insensitive* mode.

Value: Specify the second value for the comparison. All data types are available. For example, you can select to compare the value of one LDAP attribute to the value of another LDAP attribute. Only you can determine if such a comparison is meaningful.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.10 LDAP OU Condition

The LDAP OU condition allows you to compare the DN of an OU against the DN that was used when the user authenticated. If the user's DN contains the OU, the condition matches.

LDAP OU: Select *[Current]*.

Comparison: Specify how you want the values compared. Select one of the following:

- **Contains:** Specifies that you want the condition to determine whether the user is contained by a specified organizational unit.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type.

- **Contains:** Select whether the user must be contained in the specified OU (*One Level)* or whether the user can be contained in the specified OU or a child container (*Subtree)*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the second value for the comparison. If you select *LDAP OU > Name of Identity Server Configuration > User Store Name*, you can browse to the name of the OU.

If you have more than 250 OUs defined in your tree, you are prompted to enter an LDAP query string. In the text box, you need to add only the <strFilter> value for the query. For example:

<strFilter> Value	Description
admin*	Returns all OUs that begin with admin, such as adminPR, adminBG, and adminWTH.
*test	Returns all OUs that end with test, such as doctest, softtest, and securtest.
low	Returns all OUs that have "low" in the name, such as low, yellow, and clowns.

For more information about the <strFilter> parameter, see RFC 2254 "LDAP Search Filter."

If you select *Data Entry Field,* you can enter the DN of the OU in the text field. For example:

```
cn=users,dc=bcf2,dc=provo,dc=novell,dc=com
```

```
ou=users,o=novell
```

If you have defined a Liberty User Profile or an LDAP attribute for the OU you want to match, select this option, then select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.11 Liberty User Profile Condition

The Liberty User Profile condition allows you to restrict access based on a value in a Liberty User Profile attribute. The Liberty attributes must be enabled before you can use them in policies (click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Server Provider*, then enable one or more of the following: *Employee Profile*, *Personal Profile*).

These attributes can be mapped to LDAP attributes (click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *LDAP Attribute Mapping*). When mapped, the actual value comes from your user store. If you are using multiple user stores with different LDAP schemas, mapping similar attributes to the same Liberty User Profile attribute allows you to create one policy with the Liberty User Profile attribute rather than multiple policies for each LDAP attribute.

The selected attribute is compared to a value of the following type:

- Roles from an identity provider
- Date and time and its various elements
- URL and its various elements
- IP address
- Authentication contract
- Credential profile
- HTTP request method
- LDAP attribute
- Static value in a data entry field

To set up the matching for this condition, fill in the following fields:

Liberty User Profile: Select the Liberty User Profile attribute. These attributes are organized into two main groups: Corporate Employment Identity and Entire Personal Identity. By default, the Common Last Name attribute for Liberty User Profile is mapped to the sn attribute for LDAP. To select this attribute for comparison, click *Entire Personal Identity* > *Entire Common Name* > *Common Analyzed Name* > *Common Last Name*.

Comparison: Select the comparison type that matches the data type of the selected attribute and the value.

Mode: Select the mode, if available, that matches the data type. For example, if you select to compare the values as strings, you can select either a *Case Sensitive* mode or a *Case Insensitive* mode.

Value: Select one of the values that is available from the current request or select *Data Entry Field* to enter a static value. The static value that you can enter is dependent upon the comparison type you selected.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.12 Roles Condition

If you have configured some Access Manager role policies (see Section 2.2, "Creating Roles," on page 27), you can use these roles as conditions to control access. Roles are not assigned to users until the users authenticate. All authenticated users are assigned the `authenticated` role. If you use a comparison type of starts with, ends with, or contains substring, carefully evaluate the potential results. For example, if you specify `ed` as the value for an ends with comparison, the condition matches roles such as `contracted` and `assigned` that you created, but it also matches the `authenticated` role.

Fill in the following fields:

Roles: Select the role. To compare the roles the user is currently assigned with a specific role, select [Current].

Comparison: Select one of the following types:

- **Comparison: String:** Specifies that you want the values compared as strings, and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the values must match, letter for letter.
  - **Starts with:** Indicates that the Roles value must begin with the letters specified in the *Value* field.
  - **Ends with:** Indicates that the Roles value must end with the letters specified in the *Value* field.
  - **Contains Substring:** Indicates that the Roles value must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: If you have created Identity Server roles policies, select *Roles*, then select the role you want the user to have to match this condition. The `authenticated` role is assigned to all users when they authenticate. If you have defined a Liberty User Profile or an LDAP attribute for a role, you can select this option, then select your attribute.

You can use the *Data Entry Field* option to enter the name of the role you want to test for. If you have activated roles from an external source, use this option to specify the name of the role.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.13 URL Condition

The URL condition allows you to restrict access based on the URL specified in the request. If you have users requesting a resource with a URL you don't want them to use, you can use this condition in an Access Gateway Authorization policy to deny them access to this URL, and use the Actions section to redirect the request to the URL you want them to use. In a J2EE Agent policy, you can only deny or allow; you cannot redirect.

To set up matching for this condition, fill in the following fields:

Comparison: Specify how the URL is compared to the data in the *Value* field. Select one of the following types:

- **Comparison: URL: Equals:** Specifies that you want the values compared as URLs.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: URL: Equals:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

  The URL query strings can also be used in comparisons. Select Regular Expression Match to implement it. For example, if you want to match against a query string parameter xyz, whose value is abc, you need to enter the following regular expression in the *Data Entry* field:

  ```
  ".*\?.*xyz=abc.*"
  ```

  This follows the java regular expression pattern. For more information, see *The API specification for the Java 2 Platform (http://docs.oracle.com/javase/1.4.2/docs/api/java/util/regex/Pattern.html)* for the Java Regular Expression syntax.

Value: To enter a static value to compare to the URL in the current request, select *Data Entry Field* and specify the URL. This should be the complete URL, starting with the URL scheme (http:// or https://) and including the domain name, but not the port. If the URL contains a path, you must include it. If you do not specify a scheme, HTTP is used.

If you selected *Regular Expression: Matches*, regular expression rules apply.

If you selected *URL: Equals* for your comparison type, the wildcard characters (?) or (*) can be specified as the last element of the URL path to aid in matching basic URL patterns. These wildcard characters are interpreted as follows:

- ? matches all files at the specified directory level
- * matches all files and directories at and beyond the specified directory level

For example, if the request URL is `http://www.resourcehost.com/path/resource.gif`, the following entered URLs would match the request URL:

```
http://www.resourcehost.com/path/resource.gif
http://www.resourcehost.com/path/?
http://www.resourcehost.com/path/*
http://www.resourcehost.com/*
```

If you selected *URL:Equals* for the comparison type, you can add multiple values:

- Use the *Edit* button to access a text box where you can enter multiple values, each on a separate line. For more information, see "Edit Button" on page 112.
- Use the *Add* button to add values one at a time.
- Use the *URL Dredge* button to display a list of links to use as values. For more information about this option, see Using the URL Dredge Option.

All entered URLs are compared to the request URL until a match is found or the list is exhausted.

If you have defined a Liberty User Profile or an LDAP attribute for a URL, you can select these options for the value type, then select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.14 URL Scheme Condition

The URL Scheme condition allows you to restrict access based on the scheme specified in the URL of the request. For example in an Access Gateway Authorization policy, if the request contains HTTP as the scheme in the URL and you require users to use HTTPS, you can use this condition to deny access and redirect them to another URL. In a J2EE Agent policy, you can only deny or allow; you cannot redirect.

This condition allows you to compare A to B or B to A. You need to decide whether you want to compare a static value to the current value in the HTTP request, or whether you want to compare the current value in the HTTP request to a specified value. The comparison type you use depends upon the value you want to specify. If you want more flexibility in specifying the value, you should select to compare the current value in the HTTP request with a specified value.

To set up matching for this condition, fill in the following fields:

URL Scheme: Specify the scheme you want compared. You can select *Current* for the current value in the HTTP request, or specify a static value of *http* or *https*.

Comparison: Select one of the following types:

- **Comparison: URL Scheme:** Specifies that you want the values compared as scheme strings and how you want the values compared. Select one of the following:
  - **Equals:** Indicates that the URL scheme must contain the same letters, in the same order as specified in the value.

- **Starts with:** Indicates that the URL scheme must begin with the letters specified in the value.
- **Ends with:** Indicates that the URL scheme must end with the letters specified in the value.
- **Contains Substring:** Indicates that the URL scheme must contain the letters, in the same sequence, as specified in the *Value* field.

- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: String:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value you want to compare with the URL Scheme value. If you select a static value for the URL Scheme value, select *URL Scheme* and *Current*. If you select *Current* for the URL Scheme value, select one of the following value types:

- **Data Entry Field:** Allows you to specify the scheme value you want to use in the comparison. The scheme cannot be specified with a trailing colon (:) character and must be specified in lowercase (*http* or *https*). Use the *Edit* button to access a text box where you can enter multiple values, each on a separate line. (For more information, see Section 3.6.23, "Edit Button," on page 112.) Use the *Add* button to add values one at a time.

  All entered URL schemes are compared to the requested URL scheme until a match is found or the list is exhausted.
- **LDAP Attribute:** If you have defined an LDAP attribute containing a URL or URL scheme, you can select this option, then select your attribute.
- **Liberty User Profile:** If you have defined a Liberty User Profile attribute containing a URL or URL scheme, you can select this option, then select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.15 URL Host Condition

The URL Host condition allows you to restrict access based on the hostname specified in the URL of the request. For example, you can use this condition to create rules that allow access if the URL contains one hostname, but deny access if the URL contains another hostname. The URL Host condition compares the hostname in the URL of the current request to the URL hostname specified in the *Value* field.

To set up matching for this condition, fill in the following fields:

Comparison: Specify how the URL Host is compared to the data in the *Value* field. Select one of the following types:

- **Comparison: URL Host: Equals:** Specifies that you want the values compared as hostnames.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions. If you select this option, you must also specify a *Mode*. Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Select one of the following value types, then specify a value:

- **Data Entry Field:** To specify a static value to compare to the URL host in the current request, select this value type and specify the DNS name of the host.

  For example, if the request URL is http://www.resourcehost.com/path/resource.gif, the following hostname matches the resource URL:

  ```
  www.resourcehost.com
  ```

  If you selected *URL Host:Equals* for the comparison type, you can add multiple values:

  - Use the *Edit* button to access a text box where you can enter multiple values, each on a separate line. For more information, see Section 3.6.23, "Edit Button," on page 112.
  - Use the *Add* button to add values one at a time.

  All listed hostnames are compared to the requested URL until a match is found or the list is exhausted.
- **LDAP Attribute:** If you have defined an LDAP attribute containing a URL or URL host, you can select this option, then select your attribute.
- **Liberty User Profile:** If you have defined a Liberty User Profile attribute containing a URL or URL host, you can select this option, then select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.16 URL Path Condition

The URL Path condition allows you to restrict access based on the path specified in the URL of the request. This condition compares the path of the URL in the current request to the path specified in the *Value* field.

To set up matching for this condition, fill in the following fields:

Comparison: Select one of the following types:

- **Comparison: URL Path:** Specifies that you want the values compared as paths and how you want the string values compared. Select one of the following:
  - **Equals:** Indicates that the URL path must contain the same letters, in the same order as specified in the value.
  - **Starts with:** Indicates that the URL path must begin with the letters specified in the value.
  - **Ends with:** Indicates that the URL path must end with the letters specified in the value.
  - **Contains Substring:** Indicates that the URL path must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: URL Path:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value type and value for the comparison. Select one of the following:

- **Data Entry Field:** To enter a static value to compare to the URL path in the current request, select this value type and specify the path. Start the path with a forward slash.

  IMPORTANT: If you need to add a space in the path, you need to enter this encoded value for the space: %20

  If you have selected *Regular Expression: Matches* for your comparison type, regular expression rules apply. If you have selected *URL Path* for your comparison type, the path can end with a filename or a wildcard. An asterisk (*) matches all files and directories at and beyond the specified directory level. A question mark (?) matches all files at the specified directory level. For example:

Path	Match Description
`/path1/path2/`	Requires an exact match of the URL path. It matches if the URL does not contain anything after `path2`.
`/path1/file.ext`	Requires an exact match of the URL path, including the extension on the filename.
`/path1/path2/?`	Matches everything that immediately follows `path2`. It does not match anything if the path contains another directory, such as `/path1/path2/path3/file3.ext`.
`/path1/path2/*`	Matches everything that follows `path2`, including a filename or another directory, such as `/path1/path2/path3/file3.ext`.

If you selected *URL Path* for the comparison type, you can add multiple values:

- Use the *Edit* button to access a text box where you can enter multiple values, each on a separate line. For more information, see Section 3.6.23, "Edit Button," on page 112.
- Use the *Add* button to add values one at a time.

All entered URL paths are compared to the request URL path until a match is found or the list is exhausted.

- **LDAP Attribute:** If you have defined an LDAP attribute containing a URL or URL path, you can select this option, then select your attribute.
- **Liberty User Profile:** If you have defined a Liberty User Profile attribute containing a URL or URL path, you can select this option, then select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.17 URL File Name Condition

The URL File Name condition allows you to restrict access based on the filename specified in the URL. It compares the filename in the URL of the current request to the filename specified in the *Value* field.

To set up matching for this condition, fill in the following fields:

Comparison: Select one of the following types:

- **Comparison: URL File:** Specifies that you want the values compared as filenames and how you want the names compared. Select one of the following:
  - **Equals:** Indicates that the filenames must contain the same letters, in the same order as specified in the value.
  - **Starts with:** Indicates that the filenames must begin with the letters specified in the value.
  - **Ends with:** Indicates that the filenames must end with the letters specified in the value.
  - **Contains Substring:** Indicates that the filenames must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: URL File:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value type and value for the comparison. Select one of the following:

- **Data Entry Field:** To specify a static value to compare to the filename in the current request, select this value type and specify the filename.

  The value you specify is compared to what follows the last slash in the URL. If you selected *Regular Expression: Matches* for your comparison type, regular expression rules apply. If you selected *URL File* for your comparison type, enter a value that matches your string comparison type. Do not use wildcards in your value.

  If you selected *URL File* for the comparison type, you can add multiple values:

  - Use the *Edit* button to access a text box where you can enter multiple values, each on a separate line. For more information, see Section 3.6.23, "Edit Button," on page 112.
  - Use the *Add* button to add values one at a time.

  All listed filenames are compared to the requested URL filename until a match is found or the list is exhausted.
- **LDAP Attribute:** If you have defined an LDAP attribute containing a URL or filename, you can select this option, then select your attribute.
- **Liberty User Profile:** If you have defined a Liberty User Profile attribute containing a URL or filename, you can select this option, then select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.18 URL File Extension Condition

The URL File Extension condition allows you to restrict access based on the file extension specified in the URL of the request. It compares the file extension in the URL of the current request to the extension specified in the *Value* field.

To set up matching for this condition, fill in the following fields:

Comparison: Select one of the following types:

- **Comparison: URL File:** Specifies that you want the values compared as file extensions and how you want the file extensions compared. Select one of the following:
  - **Equals:** Indicates that the file extensions must contain the same letters, in the same order as specified in the value.
  - **Starts with:** Indicates that the file extensions must begin with the letters specified in the value.
  - **Ends with:** Indicates that the file extensions must end with the letters specified in the value.
  - **Contains Substring:** Indicates that the file extensions must contain the letters, in the same sequence, as specified in the *Value* field.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions.

Mode: Select the mode appropriate for the comparison type:

- **Comparison: URL File Extension:** Specify whether case is important by selecting *Case Sensitive* or *Case Insensitive*.
- **Comparison: Regular Expression: Matches:** Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value type and value for the comparison. Select one of the following:

- **Data Entry Field:** To specify a static value to compare to the file extension in the current request, select this value type and specify the file extension. You can specify the extension or the period and the extension. For example:

  ```
  .ext
  ext
  ```

  This condition does not support wildcards. If you selected *URL File Extension* for the comparison type, you can add multiple values:
  - Use the *Edit* button to access a text box where you can enter multiple values, each on a separate line. For more information, see Section 3.6.23, "Edit Button," on page 112.
  - Use the *Add* button to add values one at a time.

  All entered URL file extensions are compared to the requested URL file extension until a match is found or the list is exhausted.
- **LDAP Attribute:** If you have defined an LDAP attribute containing a URL or file extension, you can select this option, then select your attribute.
- **Liberty User Profile:** If you have defined a Liberty User Profile attribute containing a URL or file extension, you can select this option, then select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.19 X-Forward-For IP Condition

For added security, you can add the IP address of the reverse proxy as a condition to check before granting access. One way to implement this is to create a rule that requires the X-Forwarded-For IP address in the HTTP header to match the configured IP address of the reverse proxy that is using the policy. The X-Forwarded-For IP condition matches the first IP address in the X-Forwarded-For header with the IP address specified in the *Value* field.

To set up matching for this condition, fill in the following fields:

Comparison: Specify how the X-Forwarded-For IP address is compared to the data in the *Value* field. Select one of the following types:

- **Comparison: IP:** Specifies that you want the values compared as IP addresses. Select one of the following:
  - **Equals:** Allows you to specify an IP address that the X-Forwarded-For IP address must match. You can specify more than one.
  - **In Range:** Allows you to specify a range of IP addresses that the X-Forwarded-For IP address must fall within. You can specify more than one range.
  - **In Subnet:** Allows you to specify the subnet that the X-Forwarded-For IP address must belong to. You can specify more than one subnet.
- **Comparison: Regular Expression: Matches:** Specifies that you want the values compared as regular expressions. If you select this option, you must also specify a *Mode*. Select one or more of the following:

  Canonical Equivalence
  Case Insensitive
  Comments
  Dot All
  Multi-Line
  Unicode
  Unix Lines

  For regular expression syntax information, see the Javadoc for java.util.regex.Pattern.

Value: Specify the value type and value for the comparison. Select one of the following:

- **Data Entry Field:** To specify a static value, select *Data Entry Field* and provide a value appropriate for your comparison type. For example:

Comparison Type	Value
Equals	`10.10.10.10` `10.10.10.11`
In Range	`10.10.10.10 - 10.10.10.100` `10.10.20.10 - 10.10.20.100`
In Subnet	`10.10.10.12 / 22` `10.10.20.30 / 22`

If you selected *IP* for the comparison type, you can add multiple values:

- Use the *Edit* button to access a text box where you can enter multiple values, each on a separate line.
- Use the *Add* button to add values one at a time.

All listed values are compared to the IP address in the header until a match is found or the list is exhausted.

- **Client IP:** If you want the first IP address in the X-Forwarded-For header compared to the IP address of the client making the request, select this option.
- **LDAP Attribute:** If you have defined an LDAP attribute for an IP address, you can select this option, then select your attribute.
- **Liberty User Profile:** If you have defined a Liberty User Profile attribute for an IP address, you can select this option, then select your attribute.

Result on Condition Error: Specify what the condition returns when the comparison of the two values returns an error rather than the results of the comparison. Select either *False* or *True*. If you do not want the action applied when an error occurs, select *False*. If you want the action applied when an error occurs, select *True*.

3.6.20 Condition Extension

If you have loaded and configured an authorization condition extension, this option specifies a condition that is evaluated by an outside source. This outside source returns either true or false. See the documentation that came with the extension for information about what is evaluated.

3.6.21 Data Extension

If you have loaded and configured an authorization data extension, this option specifies the value that the extension retrieves. You can then select to compare this value with an LDAP attribute, a Liberty User Profile attribute, a Data Entry Field, or another Data Extension. For more information, see the documentation that came with the extension.

3.6.22 Using the URL Dredge Option

In the *URL to Dredge* text box, enter the URL of a page on a Web server, then click *Display URL List*. A list of links and images appears.

For example, if you enter `www.novell.com/documentation/index.html` for the *URL to Dredge*, links such as the following appear in the *Links* section of the *URL Results* list:

```
www.novell.com/company/careers/index.html
www.novell.com/company/strategy.html
www.novell.com/documentation/novellaccessmanager/index.html
www.novell.com/documentation/novellaccessmanager31/index.html
```

Depending upon how you have configured your Web site, you need to enter either a target page or just the URL of the site to generate a list of links.

To add all links as values to the URL condition, click *Links*. To add links selectively as a value, select the check box next to each name. To dredge a link in the list, click the link.

If the URL contains images, a list of images appears in the *Images* section. To add an image as a value, select the check box next to the image name.

To save your changes, click *OK*.

IMPORTANT: If you attempt to dredge an HTTPS site that is using a self-signed certificate, you need to import the trusted root of the site into the Trusted Roots store of the Access Gateway before performing the dredge.

3.6.23 Edit Button

Some of the conditions such as Client IP and URL display an *Edit* button when you select *Equals* as the condition and *Data Entry Field* as the value. The *Edit* button displays a text box where you can specify multiple values.

In the text box, enter each value on a separate line.

To save your modifications, click *OK*.

To discard your modifications, click *Cancel*.

3.7 Importing and Exporting Authorization Policies

You can import and export Authorization policies in order to run them in other Access Manager configurations and to analyze the authorization logic. The policy is exported as a text file with XML tags. We do not recommend editing the exported file with a text editor. Any changes you want to make to a policy should be done through the Administration Console.

To export an Authorization policy:

1. In the Administration Console, click *Policies* > *Policies*.
2. Select an Authorization policy, then click *Export*.
3. (Optional) Modify the name suggested for the file.
4. Click *OK*.
5. Using the features of your browser, specify where you want the file to be copied.
6. Click *OK*.

To import a policy:

1. Make sure any referenced Role policies have been imported.

   See Section 2.7, "Importing and Exporting Role Policies," on page 63.
2. If the policy uses LDAP or Liberty Profile attributes, make sure the Identity Server has been configured for these same attributes.
3. In the Administration Console, click *Policies* > *Policies*.
4. Click *Import*, then browse to and select the file.
5. Click *OK*.
6. When the policy appears in the list, click *Apply Changes*.

4 Creating Identity Injection Policies

Identity injection allows you to add information to the URL or to the HTML page before it is posted to the Web server. The Web server uses this information to determine whether the user should have access to the resource, so it is the Web server that determines the information that you need to inject to allow access to the resource.

Identity injection is one of the features of Access Manager that enable you to provide single sign-on for your users. When the policy is configured correctly, the user is unaware that additional information is required to access a Web server.

IMPORTANT: Identity Injection policies allow you to inject the user's password into the HTTP header. If you set up such a policy, you should also configure the Access Gateway to use SSL between itself and the back-end Web server. This is the only way to ensure that the password is encrypted on the wire.

This section describes the elements available for an Identity Injection policy, but your Web servers determine which elements you use.

- Section 4.1, "Designing an Identity Injection Policy," on page 113
- Section 4.2, "Configuring an Identity Injection Policy," on page 115
- Section 4.3, "Configuring an Authentication Header Policy," on page 116
- Section 4.4, "Configuring a Custom Header Policy," on page 120
- Section 4.5, "Configuring a Custom Header with Tags," on page 123
- Section 4.6, "Specifying a Query String for Injection," on page 125
- Section 4.7, "Injecting into the Cookie Header," on page 128
- Section 4.8, "Importing and Exporting Identity Injection Policies," on page 128
- Section 4.9, "Sample Identity Injection Policy," on page 129

4.1 Designing an Identity Injection Policy

Before setting up an Identity Injection policy, you need to know the following about your Web application:

- Does it require an authentication header? Does this header need just the username or does it also need the password?
- Does it use a custom header with custom names (x-names)? If so, you need to know their names and their expected values.
- Does the custom header require any custom names (x-names) with tags? If so, gather this information.
- Does the application expect specific values in the query string of the URL? If so, gather this information.

After gathering the information, you need to determine whether you need to create one policy with one rule, one policy with multiple rules, or multiple policies. If you have multiple applications that require the same type of authentication header, you might want to create an authentication header policy and separate policies for the application-specific information. You can then enable both the authentication header policy and the application-specific policy for the resource that is protecting the application. You should design your policies so that the application receives just what it needs. It should not inject custom names and values it does not use.

Everything defined in a policy is injected into the header, even if the values are empty because the Access Manager could not obtain the value for the item. For some applications, this is still useful information and the application uses it to make access decisions.

Whether you create a policy with one rule or multiple rules is a personal design decision. If you put all the actions in one rule, you have only one description field to describe the function of the policy. If you put each action type in a separate rule, you have multiple description fields to describe the function of the policy. Select the method that is easiest for you.

Rules are evaluated by priority. The first rule that is evaluated with an authentication header is processed, and the authentication header is rejected if it is found in any of the other rules. Your policy can inject only one authentication header, one cookie header, and one query string, but it can inject multiple custom headers and custom headers with tags.

4.1.1 Using the Refresh Data Option

Identity Injection policies are processed when a user requests access to a resource. The results and the values of the data items are cached for the user session. This means that when the user requests a second time to access the resource, the policy is evaluated, but the data values from the first evaluation are used. When a data item is cached for the user session, the user must log out and log back in to trigger new data values. (For information on how long the data items are cached, see Section 7.4, “The Policy Is Using Old User Data,” on page 182.)

The LDAP Attribute and the Shared Secret actions can be configured to refresh their values. This means the attribute or secret value is read not just on the first request that triggers the policy evaluation, but when the specified refresh interval expires. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes.

You can use this feature for situations when you do not want to force the user to log in again to gain rights to resources or to revoke rights to resources. For example, suppose that you have an Identity Injection policy that grants access based on an LDAP attribute in a custom header having a “yes” value. Users with a “no” value in custom header are denied access.

If you don’t enable the Refresh Data option on this attribute in the policy, the policy is evaluated when the user first tries to access the resource. The value for the attribute is cached for the user session, and until the user logs out, that is the value that is used.

However, if you enable the Refresh Data option on this attribute in the policy, the policy is evaluated when the user first tries to access the resource. When the user sends a second request to access the resource and the specified interval has expired, the Refresh Data option causes the value of the attribute to be read again from the LDAP server. This new value is injected into the custom header, and any other policy that is triggered by the request and uses the new value for its policy.

- If the value from the first request to the second request changes from no to yes, the user gets access to the resource.
- If the value from the first request to the second request changes from yes to no, the user is denied access to the resource.

For example:

- If the attribute controls access to employee resources and an employee leaves, a quick change of this attribute value cuts the employee off from the resources that should be available to employees only.
- If the attribute controls access to a software download site and a user has just purchased a product, a quick change to this attribute value can grant access to the download site.

IMPORTANT: This feature needs to be used with caution. Because querying the LDAP server slows down the processing of a policy, LDAP attribute and secret store values are normally cached for the user session. Enable this option only on those attributes and secrets that are critical to the security of your system or to the design of your work flow.

4.2 Configuring an Identity Injection Policy

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New.*
3. Specify a name for the policy, select *Access Gateway: Identity Injection* for the type of policy, then click *OK*.

4. Fill in the following fields:

   Description: (Optional) Describe the purpose of this policy. Because Identity Injection policies are customized to match the content of a specific Web server, you might want to include the name of the Web server as part of the description.

   Priority: Specify the order in which a rule is applied in the policy, when the policy has multiple rules. The highest priority is 1 and the lowest priority is 10.

5. In the *Actions* section, click *New*, then select one of the following.
   - **Inject into Authentication Header:** Inserts the username and password into the header. For information about how to configure this type of policy, see Section 4.3, "Configuring an Authentication Header Policy," on page 116.
   - **Inject into Custom Header:** Inserts custom names with values into the custom header. For information about how to configure this type of policy, see Section 4.4, "Configuring a Custom Header Policy," on page 120.

- **Inject into Custom Header with Tags:** Inserts custom tags with name/value content into the custom header. For information about how to configure this type of policy, see Section 4.5, "Configuring a Custom Header with Tags," on page 123.
- **Inject into Query String:** Inserts a query string into the URL for the page. For information about how to configure this type of policy, see Section 4.6, "Specifying a Query String for Injection," on page 125.
- **Inject into Cookie Header:** Inserts the session cookie into the cookie header. For information about how to configure this type of policy, see Section 4.7, "Injecting into the Cookie Header," on page 128.

6 (Optional) Repeat Step 5.

Repeat this process to add multiple actions to the same rule. If a particular action is allowed only once per rule, then the action does not appear in the *New* menu if that action has already been defined in the rule. If an action is allowed multiple times per rule, you can select it from the *New* menu or use the *Copy Action* icon and modify the new entry.

7 To save the policy, click *OK* twice, then click *Apply Changes*.

8 For information on how to assign the policy to a protected resource, see "Assigning an Identity Injection Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.

4.3 Configuring an Authentication Header Policy

To inject values into the authentication header, you need to know what the Web server requires. For basic authentication, you need to inject the username and password. For a sample policy for a Web server that requires the LDAP username and password to be injected into the header, see "Setting Up an Identity Injection Policy" in the *NetIQ Access Manager 3.2 Setup Guide*.

To create and configure an authentication header policy:

1 In the Administration Console, click *Policies* > *Policies*.

2 Select the policy container, then click *New*.

3 Specify a name for the policy, select *Access Gateway: Identity Injection* for the type, then click *OK*.

4 (Optional) Specify a description for the injection policy. This is useful if you plan to create multiple policies to be used by multiple resources.

5 In the *Actions* section, click *New*, then select *Inject into Authentication Header*.

6 Fill in the *User Name* field.

Select *Credential Profile* to insert the name the user entered when the user authenticated. This is the most common value type to use for the username.

The default contracts assign the cn attribute to the Credential Profile. If you have created a custom contract that uses credentials other than the ones listed below, do not use the Credential Profile as a condition.

If your user store is an Active Directory server, the SAMAccountName attribute is used for the username and stored in the cn field of the LDAP Credential Profile.

Depending upon what the user must supply for authentication, select one of the following:

- **LDAP Credentials:** If you prompt the user for a username, select this option, then select either *LDAP User Name* (the cn attribute of the user) or *LDAP User DN* (the fully distinguished name of the user). Your Web server requirements determine which one you use.
- **X509 Credentials:** If you prompt the user for a certificate, select this option, then select one of the following options depending upon your Web server requirements.
  - **X509 Public Certificate Subject:** Injects just the subject field from the certificate, which can match the DN of the user, depending upon who issued the certificate.
  - **X509 Public Certificate Issuer:** Injects just the issuer field from the certificate, which is the name of the certificate authority (CA) that issued the certificate.
  - **X509 Public Certificate:** Injects the entire certificate.
  - **X509 Serial Number:** Injects the certificate serial number.
- **SAML Credential:** Although this option is available for the username, most applications that use SAML assertions use them for the user's password. For the username, you should probably select an option that allows you to supply the user's name, such as *LDAP Credentials* or *LDAP Attribute*.

Your Web server requirements determine the data type you select for the username. LDAP, X509, and SAML credentials are available from the Credential Profile. If you have created a custom contract that uses a credential other than the ones listed in the Credential Profile, you can select one of the following values to insert into the header as the username:

- **Authentication Contract:** Injects the URI of the authentication contract the user used for authentication.
- **Client IP:** Injects the IP address associated with the user.
- **LDAP Attribute:** Injects the value of the selected attribute. For Active Directory servers, specify the SAMAccountName attribute for the username. If the attribute you require does not appear in the list, click *New LDAP Attribute* to add the attribute.

  The *Refresh Data Every* option allows you to determine when to send a query to the LDAP server to verify the current value of the attribute. Because querying the LDAP server slows down the processing of a policy, LDAP attribute values are normally cached for the user session.

  Change the value of this option from session to a more frequent interval only on those attributes that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes.

  For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.
- **Liberty User Profile:** Injects the value of the selected attribute. If no profile attributes are available, you have not enabled their use in the Identity Server configuration. See "Managing Web Services and Profiles" in the *NetIQ Access Manager 3.2 Identity Server Guide*.
- **Proxy Session Cookie:** Injects the session cookie associated with the user.
- **Roles:** Injects the roles that have been assigned to the user.
- **Shared Secret:** Injects the username that has been stored in the selected shared secret store.

You can create your own username attribute. Click *New Shared Secret*, specify a display name for the store, and Access Manager creates the store. Select the store, click *New Shared Secret Entry*, specify a name for the attribute, then click *OK*. The store can contain one name/value pair or a collection of name/value pairs. For more information, see Section 5.4, "Creating and Managing Shared Secrets," on page 150.

The *Refresh Data Every* option allows you to determine when to send a query to verify the current value of the secret. Because querying slows down the processing of a policy, secret values are normally cached for the user session.

Change the value of this option from session to a more frequent interval only on those secrets that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes. For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.

- **String Constant:** Injects a static value that you specify in the text box. This value is used by all users who access the resources assigned to this policy.
- **Java Data Injection Module:** Specifies the name of a custom Java plug-in, which injects custom values into the header. Usually, you can use either the *LDAP Attribute* or *Liberty User Profile* option to supply custom values, because both are extensible. For more information about creating a custom plug-in, see *Novell® Access Manager Developer Tools and Examples* (http://www.novell.com/developer/).
- **Data Extension:** (Conditional) If you have installed a data extension for Identity Injection policies, this option injects the value that the extension retrieves. For more information about creating a data extension, see *Novell® Access Manager Developer Tools and Examples* (http://www.novell.com/developer/).

The value type you use depends upon how you have set up the application.

NOTE: To improve the policy's performance, configure the LDAP Attributes, Credential Profile, Liberty User Profile, and Shared Secret attributes to be sent with authentication. For more information, see "Configuring the Attributes Sent with Authentication"in the *NetIQ Access Manager 3.2 Identity Server Guide*.

---

7 Fill in the *Password* field.

Select *Credential Profile* to insert the password the user entered when the user authenticated. This is the most common value type to use for the password. If you have created a custom contract that uses credentials other than the ones listed below for the password, do not use the Credential Profile for the password.

- **LDAP Credentials:** If you prompt the user for a password, select this option, then select *LDAP Password*. If the user's password is the same as the name of the user, you can select either *LDAP User Name* (the cn attribute of the user) or *LDAP User DN* (the fully distinguished name of the user).
- **X509 Credentials:** If you use a certificate for the password, select this option, then select one of the following:
  - **X509 Public Certificate Subject:** Injects just the subject from the certificate, which can match the DN of the user, depending upon who issued the certificate.
  - **X509 Public Certificate Issuer:** Injects just the issuer from the certificate, which is the name of the certificate authority (CA) that issued the certificate.
  - **X509 Public Certificate:** Injects the entire certificate.
  - **X509 Serial Number:** Injects the certificate serial number.
- **SAML Credential:** Injects the SAML assertion in the authentication header as the user's password.

Your Web server requirements determine the data type you select for the password. LDAP, X509, and SAML credentials are available from the Credential Profile. You can also select one of the following values to insert into the header as the password:

- **Authentication Contract:** Injects the URI of a local authentication contract that the user used for authentication.
- **Client IP:** Injects the IP address associated with the user.
- **LDAP Attribute:** Injects the value of the selected attribute. For Active Directory servers, specify the SAMAccountName attribute for the username. If the attribute you require does not appear in the list, click *New LDAP Attribute* to add the attribute.

  The *Refresh Data Every* option allows you to determine when to send a query to the LDAP server to verify the current value of the attribute. Because querying the LDAP server slows down the processing of a policy, LDAP attribute values are normally cached for the user session.

  Change the value of this option from session to a more frequent interval only on those attributes that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes. For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.
- **Liberty User Profile:** Injects the value of the selected attribute.
- **Proxy Session Cookie:** Injects the session cookie associated with the user.
- **Roles:** Injects the roles that have been assigned to the user.
- **Shared Secret:** Injects the password that has been stored in the selected shared secret store.

  You can create your own password attribute. Click *New Shared Secret*, specify a display name for the store, and the Access Manager creates the store. Select the store, click *New Shared Secret Entry*, specify a name for the attribute, then click *OK*. The store can contain one name/value pair or a collection of name/value pairs. For more information, see Section 5.4, "Creating and Managing Shared Secrets," on page 150.

  The *Refresh Data Every* option allows you to determine when to send a query to verify the current value of the secret. Because querying slows down the processing of a policy, secret values are normally cached for the user session.

  Change the value of this option from session to a more frequent interval only on those secrets that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes. For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.
- **String Constant:** Injects a static value that you specify in the text box. This value is used by all users who access the resources assigned to this policy.
- **Java Data Injection Module:** Specifies the name of a custom Java plug-in, which injects custom values into the header. Usually, you can use either the *LDAP Attribute* or *Liberty User Profile* option to supply custom values, because both are extensible. For more information about creating a custom plug-in, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).
- **Data Extension:** (Conditional) If you have installed a data extension for Identity Injection policies, this option injects the value that the extension retrieves. For more information about creating a data extension, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).

The value type you use depends upon how you have set up the application.

8 Specify the format for the value:

Multi-Value Separator: Select a value separator, if the value type you have select is multi-valued. For example, *Roles* can contain multiple values.

DN Format: If the value is a DN, select the format for the DN:

- **LDAP:** Specifies LDAP typed comma notation:

  `cn=jsmith,ou=Sales,o=novell`

- **NDAP Partial Dot Notation:** Specifies eDirectory™ typeless dot notation.

  `jsmith.sales.novell`

- **NDAP Leading Partial Dot Notation:** Specifies eDirectory typeless leading dot notation.

  `.jsmith.sales.novell`

- **NDAP Fully Qualified Partial Dot Notation:** Indicates eDirectory typed dot notation.

  `cn=jsmith.ou=Sales.o=novell`

- **NDAP Fully Qualified Leading Dot Notation:** Indicates eDirectory typed leading dot notation.

  `.cn=jsmith.ou=Sales.o=novell`

9 Click *OK*.

10 (Optional) To add a second rule, click *New* in the Rule List.

You can inject only one authentication header into an Identity Injection rule. However, your policy can have multiple rules. If you inject two authentication headers, each in a separate rule, the authentication header in the rule with the highest priority is applied, and the authentication header action in the second rule is ignored.

11 To save the policy, click *OK*, then click *Apply Changes*.

4.4 Configuring a Custom Header Policy

To inject values into a custom header, you need to know the name of the tag and its expected value type. The names are specific to the application. The names might be case sensitive. They might require an X- prefix. Because the requirements vary, you need to enter them in the format as specified by the application. For example, an application might require the following to be in the custom header:

Name/Value Pair	Description
`X-First_Name=givenName`	A first name tag with an LDAP attribute value
`X-Last_Name=sn`	A last name tag with an LDAP attribute value
`X-Role=sales_role`	A role tag with the role name as the value.

If you create a custom header policy with these name/value pairs, the policy injects these names with their values into a custom header, before sending the request to the Web server.

To create such a policy:

1 In the Administration Console, click *Policies* > *Policies*.

2 Select the policy container, then click *New.*

3 Specify a name for the policy, select *Access Gateway: Identity Injection* for the type, then click *OK*.

4 (Optional) Specify a description for the injection policy. This is useful if you plan to create multiple custom header policies to be used for multiple resources.

5 In the *Actions* section, click *New*, then select *Inject into Custom Header*.

6 Fill in the following fields:

Custom Header Name: Specify the name to be inserted into the custom header. These are the names required by your application. If your application requires the X- prefix, make sure you include the prefix in this field.

Value: Select the value required by the name. Select one of the following:

- **Authentication Contract:** Injects the URI of a local authentication contract that the user used for authentication.
- **Client IP:** Injects the IP address associated with the user.
- **Credential Profile:** Injects the credentials that the user specified at login. You can select *LDAP Credentials*, *X509 Credentials*, or *SAML Credentials*. For more information, see Section 4.3, "Configuring an Authentication Header Policy," on page 116.
- **LDAP Attribute:** Injects the value of the selected attribute. For Active Directory servers, specify the SAMAccountName attribute for the username. If the attribute you require does not appear in the list, click *New LDAP Attribute* to add the attribute.

  The *Refresh Data Every* option allows you to determine when to send a query to the LDAP server to verify the current value of the attribute. Because querying the LDAP server slows down the processing of a policy, LDAP attribute values are normally cached for the user session.

  Change the value of this option from session to a more frequent interval only on those attributes that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes.

  For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.
- **Liberty User Profile:** Injects the value of the selected attribute. If no profile attributes are available, you have not enabled their use in the Identity Server configuration. See "Managing Web Services and Profiles" in the *NetIQ Access Manager 3.2 Identity Server Guide*.
- **Proxy Session Cookie:** Injects the session cookie associated with the user.
- **Roles:** Injects the roles that have been assigned to the user.
- **Shared Secret:** Injects a value that has been stored in the selected shared secret store. Select the shared secret store and the name of the value you want injected.

  You can create your own value. Click *New Shared Secret*, specify a display name for the store, and the Access Manager creates the store. Select the store, click *New Shared Secret Entry*, specify a name for the attribute, then click *OK*. The name you select for the attribute should

match the Custom Header name. The store can contain one name/value pair or a collection of name/value pairs. For more information, see Section 5.4, "Creating and Managing Shared Secrets," on page 150.

The *Refresh Data Every* option allows you to determine when to send a query to verify the current value of the secret. Because querying slows down the processing of a policy, secret values are normally cached for the user session.

Change the value of this option from session to a more frequent interval only on those secrets that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes. For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.

- **String Constant:** Injects a static value that you specify in the text box. This value is used by all users who access the resources assigned to this policy.
- **Java Data Injection Module:** Specifies the name of a custom Java plug-in, which injects custom values into the header. Usually, you can use either the *LDAP Attribute* or *Liberty User Profile* option to supply custom values, because both are extensible. For more information on creating a Java plug-in, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).
- **Data Extension:** (Conditional) If you have installed a data extension for Identity Injection policies, this option injects the value that the extension retrieves. For more information about creating a data extension, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).

NOTE: To improve the policy's performance, configure the LDAP Attributes, Credential Profile, Liberty User Profile, and Shared Secret attributes to be sent with authentication. For more information, see "Configuring the Attributes Sent with Authentication"in the *NetIQ Access Manager 3.2 Identity Server Guide*.

---

7 Specify the format for the value:

Multi-Value Separator: Select a value separator, if the value type you have select is multi-valued. For example, *Roles* can contain multiple values.

DN Format: If the value is a DN, select the format for the DN:

- **LDAP:** Specifies LDAP typed comma notation.

  `cn=jsmith,ou=Sales,o=novell`

- **NDAP Partial Dot Notation:** Specifies eDirectory typeless dot notation.

  `jsmith.sales.novell`

- **NDAP Leading Partial Dot Notation:** Specifies eDirectory typeless leading dot notation.

  `.jsmith.sales.novell`

- **NDAP Fully Qualified Partial Dot Notation:** Indicates eDirectory typed dot notation.

  `cn=jsmith.ou=Sales.o=novell`

- **NDAP Fully Qualified Leading Dot Notation:** Indicates eDirectory typed leading dot notation.

  `.cn=jsmith.ou=Sales.o=novell`

8 (Optional) To add additional custom header actions, click *New*, then select *Inject into Custom Header* or use the *Copy Action* icon and modify the new entry.

9 To save the policy, click *OK* twice, then click *Apply Changes*.

4.5 Configuring a Custom Header with Tags

Some Web applications require more than a name and a value to be injected into the custom header. Sometimes they require a custom name, a tag, and a value. Sometimes the application requires a custom name with multiple tags and values. The *Inject into Custom Header with Tags* option provides you with the flexibility to add such values to the custom header. For example, your application could be expecting the following custom header with tag:

```
X-Custom_Role Role=Manager
```

You can inject this information by setting the *Custom Header Name* to X-Custom, the *Tag Name* to Role, and the *Tag Value* to Manager. The value can be set as a static variable or you can retrieve it from various sources such as a Liberty User Profile attribute or the roles assigned to the current user.

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New*.
3. Specify a name for the policy, select *Access Gateway: Identity Injection* for the type, then click *OK*.
4. (Optional) Specify a description for the injection policy. This is useful if you plan to create multiple custom header policies to be used for multiple resources.
5. In the *Actions* section, click *New*, then select *Inject into Custom Header with Tags*.

6. Fill in the following fields:

   Custom Header Name: Specify the name that the application expects. If your application requires the X- prefix, make sure you include the prefix in this field.

   Tag Name: Specify the tag name that the application expects.

   Tag Value: Specify the value. Select from the following data types:

   - **Authentication Contract:** Injects the URI of a local authentication contract that the user used for authentication.
   - **Client IP:** Injects the IP address associated with the user.
   - **Credential Profile:** Injects the credentials that the user specified at login. You can select *LDAP Credentials*, *X509 Credentials*, or *SAML Credential*. For more information, see Section 4.3, "Configuring an Authentication Header Policy," on page 116.
   - **LDAP Attribute:** Injects the value of the selected attribute. For Active Directory servers, specify the SAMAccountName attribute for the username. If the attribute you require does not appear in the list, click *New LDAP Attribute* to add the attribute.

The *Refresh Data Every* option allows you to determine when to send a query to the LDAP server to verify the current value of the attribute. Because querying the LDAP server slows down the processing of a policy, LDAP attribute values are normally cached for the user session.

Change the value of this option from session to a more frequent interval only on those attributes that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes. For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.

- **Liberty User Profile:** Injects the value of the selected attribute. If no profile attributes are available, you have not enabled their use in the Identity Server configuration. See "Managing Web Services and Profiles" in the *NetIQ Access Manager 3.2 Identity Server Guide*.
- **Proxy Session Cookie:** Injects the session cookie associated with the user.
- **Roles:** Injects the roles that have been assigned to the user.
- **Shared Secret:** Injects a value that has been stored in the selected shared secret store. The name specified as the Tag Name must match the name of a name/value pair stored in the shared secret.

  You can create your own value. Click *New Shared Secret*, specify a display name for the store, and the Access Manager creates the store. Select the store, click *New Shared Secret Entry*, specify a name for the attribute, then click *OK*. The name must match the expected Tag Name. The store can contain one name/value pair or a collection of name/value pairs. For more information, see Section 5.4, "Creating and Managing Shared Secrets," on page 150.

  The *Refresh Data Every* option allows you to determine when to send a query to verify the current value of the secret. Because querying slows down the processing of a policy, secret values are normally cached for the user session.

  Change the value of this option from session to a more frequent interval only on those secrets that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes. For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.
- **String Constant:** Injects a static value that you specify in the text box. This value is used by all users who access the resources assigned to this policy.
- **Java Data Injection Module:** Specifies the name of a custom Java plug-in, which injects custom values into the header. Usually, you can use either the *LDAP Attribute* or *Liberty User Profile* option to supply custom values, because both are extensible. For more information about creating a custom plug-in, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).
- **Data Extension:** (Conditional) If you have installed a data extension for Identity Injection policies, this option injects the value that the extension retrieves. For more information about creating a data extension, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).

NOTE: To improve the policy's performance, configure the LDAP Attributes, Credential Profile, Liberty User Profile, and Shared Secret attributes to be sent with authentication. For more information, see "Configuring the Attributes Sent with Authentication"in the *NetIQ Access Manager 3.2 Identity Server Guide*.

---

7 To add multiple tag and value pairs to the custom name, click *New* in the *Tags* section.

  Use the up-arrow and down-arrow buttons to order the tags.

8 Specify the format for the value:

Multi-Value Separator: Select a value separator, if the value type you have select is multi-valued. For example, *Roles* can contain multiple values.

DN Format: If the value is a DN, select the format for the DN:

- **LDAP:** Specifies LDAP typed comma notation.

  `cn=jsmith,ou=Sales,o=novell`

- **NDAP Partial Dot Notation:** Specifies eDirectory typeless dot notation.

  `jsmith.sales.novell`

- **NDAP Leading Partial Dot Notation:** Specifies eDirectory typeless leading dot notation.

  `.jsmith.sales.novell`

- **NDAP Fully Qualified Partial Dot Notation:** Indicates eDirectory typed dot notation.

  `cn=jsmith.ou=Sales.o=novell`

- **NDAP Fully Qualified Leading Dot Notation:** Indicates eDirectory typed leading dot notation.

  `.cn=jsmith.ou=Sales.o=novell`

9. (Optional) To add additional custom header actions, click *New*, then select *Inject into Custom Header with Tags* or use the *Copy Action* icon and modify the new entry.
10. To save the policy, click *OK* twice, then click *Apply Changes*.

4.6 Specifying a Query String for Injection

Some applications require custom information in a query string of the URL. The *Inject into Query String* option allows you to inject this information without prompting the user for it. To inject the information, you must specify a tag name and a tag value. The tag name is what your application requires. For example, suppose your application expects the following query string for user jsmith:

```
?name=jsmith
```

You can inject this information into the URL by specifying a name for the *Tag Name* and *Credential Profile* for the *Tag Value*. The *Credential Profile* value type inserts the name that the current user specified when authenticating to the Access Gateway.

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New*.
3. Specify a name for the policy, select *Access Gateway: Identity Injection* for the type, then click *OK*.
4. (Optional) Specify a description for the injection policy.
5. In the *Actions* section, click *New*, then select *Inject into Query String*.

6 Fill in the following fields:

Tag Name: Specify the tag name that the application expects.

Tag Value: Specify the value. Select from the following data types:

- **Authentication Contract:** Injects the URI of a local authentication contract that the user used for authentication.
- **Client IP:** Injects the IP address associated with the user.
- **Credential Profile:** Injects the credentials that the user specified at login. You can select *LDAP Credentials*, *X509 Credentials*, or *SAML Credential*. For more information, see Section 4.3, "Configuring an Authentication Header Policy," on page 116.
- **LDAP Attribute:** Injects the value of the selected attribute. For Active Directory servers, specify the SAMAccountName attribute for the username. If the attribute you require does not appear in the list, click *New LDAP Attribute* to add the attribute.

  The *Refresh Data Every* option allows you to determine when to send a query to the LDAP server to verify the current value of the attribute. Because querying the LDAP server slows down the processing of a policy, LDAP attribute values are normally cached for the user session.

  Change the value of this option from session to a more frequent interval only on those attributes that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes. For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.
- **Liberty User Profile:** Injects the value of the selected attribute. If no profile attributes are available, you have not enabled their use in the Identity Server configuration. See "Managing Web Services and Profiles" in the *NetIQ Access Manager 3.2 Identity Server Guide*.
- **Proxy Session Cookie:** Injects the session cookie associated with the user.
- **Roles:** Injects the roles that have been assigned to the user.
- **Shared Secret:** Injects a value that has been stored in the selected shared secret store. The name specified as the Tag Name must match the name of a name/value pair stored in the shared secret.

  You can create your own value. Click *New Shared Secret*, specify a display name for the store, and the Access Manager creates the store. Select the store, click *New Shared Secret Entry*, specify a name for the attribute, then click *OK*. The name you specify must match the Tag Name. The store can contain one name/value pair or a collection of name/value pairs. For more information, see Section 5.4, "Creating and Managing Shared Secrets," on page 150.

The *Refresh Data Every* option allows you to determine when to send a query to verify the current value of the secret. Because querying slows down the processing of a policy, secret values are normally cached for the user session.

Change the value of this option from session to a more frequent interval only on those secrets that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes. For more information, see Section 4.1.1, "Using the Refresh Data Option," on page 114.

- **String Constant:** Injects a static value that you specify in the text box. This value is used by all users who access the resources assigned to this policy.
- **Java Data Injection Module:** Specifies the name of a custom Java plug-in, which injects custom values into the header. Usually, you can use either the *LDAP Attribute* or *Liberty User Profile* option to supply custom values, because both are extensible. For more information about creating a custom plug-in, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).
- **Data Extension:** (Conditional) If you have installed a data extension for Identity Injection policies, this option injects the value that the extension retrieves. For more information about creating a data extension, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).

NOTE: To improve the policy's performance, configure the LDAP Attributes, Credential Profile, Liberty User Profile, and Shared Secret attributes to be sent with authentication. For more information, see "Configuring the Attributes Sent with Authentication"in the *NetIQ Access Manager 3.2 Identity Server Guide*.

---

7 (Optional) To add multiple tag and value pairs, click *New* in the *Tags* section.

You can inject only one query string into a rule, but you can inject multiple tag-name and tag-value pairs in the single query string.

Use the up-arrow and down-arrow buttons to order the tags.

8 Specify the format for the values:

Multi-Value Separator: Select a value separator, if the value type you have select is multi-valued. For example, *Roles* can contain multiple values.

DN Format: If the value is a DN, select the format for the DN:

- **LDAP:** Specifies LDAP typed comma notation.

  ```
  cn=jsmith,ou=Sales,o=novell
  ```

- **NDAP Partial Dot Notation:** Specifies eDirectory typeless dot notation.

  ```
  jsmith.sales.novell
  ```

- **NDAP Leading Partial Dot Notation:** Specifies eDirectory typeless leading dot notation.

  ```
  .jsmith.sales.novell
  ```

- **NDAP Fully Qualified Partial Dot Notation:** Indicates eDirectory typed dot notation.

  ```
  cn=jsmith.ou=Sales.o=novell
  ```

- **NDAP Fully Qualified Leading Dot Notation:** Indicates eDirectory typed leading dot notation.

  ```
  .cn=jsmith.ou=Sales.o=novell
  ```

9 To save the policy, click *OK* twice, then click *Apply Changes*.

4.7 Injecting into the Cookie Header

Some applications require access to the Access Gateway session cookie and expect to find it in the cookie header. You can create an Identity Injection policy that adds this cookie to the cookie header.

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New*.
3. Specify a name for the policy, select *Access Gateway: Identity Injection* for the type, then click *OK*.
4. (Optional) Specify a description for the injection policy.
5. In the *Actions* section, click *New*, then select *Inject into Cookie Header*.

This action allows only one value unless you have installed a data extension. If you have installed a data extension, you can select either *Proxy Session Cookie* or the *Data Extension*.

Proxy Session Cookie: Injects the session cookie for the user.

Data Extension: Injects the value retrieved from the extension. For more information about creating a data extension, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).

6. To save the policy, click *OK* twice, then click *Apply Changes*.

4.8 Importing and Exporting Identity Injection Policies

You can import and export Identity Injection policies in order to run them in other Access Manager configurations. The policy is exported as a text file with XML tags. We do not recommend editing the exported file with a text editor. Any changes you want to make to a policy should to be done through the Administration Console.

To export an Identity Injection policy:

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container.
3. Select an Identity Injection policy, then click *Export*.
4. (Optional) Modify the name suggested for the file.
5. Click *OK*.
6. Using the features of your browser, specify where the file is to be copied.

To import a policy:

1. Make sure any referenced shared secret stores have been created. See Section 5.4, "Creating and Managing Shared Secrets," on page 150.
2. If the policy uses LDAP or Liberty Profile attributes, make sure the Identity Server has been configured for these same attributes.
3. Make sure any referenced role policies have been imported.

   See Section 2.7, "Importing and Exporting Role Policies," on page 63.

4. In the Administration Console, click *Access Manager* > *Policies*.
5. Click *Import*, then browse to the location of the file.
6. Click *OK*.
7. When the policy appears in the list, click *Apply Changes*.

4.9 Sample Identity Injection Policy

One of the common uses of an Identity Injection policy is to differentiate between internal users and external users. Web servers that have been configured for this logic can then display one set of pages to internal users and another set of pages to external users. The following sample policy is based on an environment that has the following characteristics:

- The Web server has been configured to look for a custom tag called `IPAddress` and to differentiate between internal IP addresses and external IP addresses.
- The internal customers have NAT IP addresses.
- The protected resource is a page called `mycompany.html`. This page is a public protected resource (no authentication required) because the IP address of the client is available before authentication.

To configure your site for this type of policy:

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container.
3. Click *New*, specify a name for the policy, select *Access Gateway: Identity Injection* for the type, then click *OK*.
4. In the *Actions* section, click *New* > *Inject into Custom Header*.
5. Fill in the following fields:

   Custom Header Name: Specify `IPAddress` in the text box.

   Value: Select *Client IP*.

   The other fields do not need to be modified. Your policy should look similar to the following:

6 Click *OK* twice, then click *Apply Changes*.

7 Assign the policy to the `mycompany.html` page of the Web server. Click *Access Gateways* > *Edit* > *[Name of Reverse Proxy]* > *[Name of Proxy Service]* > *Protected Resources*.

8 In the Protected Resource List, select the protected resource for the page or click *New* to create one, then specify a name for it.

9 In the *URL Path List*, ensure that the path ends with the name of the page. For example:

```
/mycompany.html
```

10 Click *Identity Injection*, select the name of the IP address policy, then click *Enable*.

11 To save the changes, click *Configuration Panel* > *OK*.

12 On the Configuration page, click *OK*, then click *Update*.

13 Configure the Web server to use the IPAddress values in the custom header to distinguish between external and internal customers.

In this sample scenario, the Web server is configured to recognize IP addresses starting with `10.` as internal customers and all other addresses as external customers.

5 Creating Form Fill Policies

A Form Fill policy allows you to prepopulate fields in a form on first login and then save the information in the completed form to a secret store for subsequent logins. The user is prompted to reenter the information only when something changes such as an expired password. Form Fill is one of the features of Access Manager that enable you to provide single sign-on for your users.

The HTML page determines the requirements for the Form Fill policy. This section describes the following:

- Section 5.1, "Understanding an HTML Form," on page 131
- Section 5.2, "Creating a Form Fill Policy for the Sample Form," on page 134
- Section 5.3, "Implementing Form Fill Policies," on page 137
- Section 5.4, "Creating and Managing Shared Secrets," on page 150
- Section 5.5, "Importing and Exporting Form Fill Policies," on page 152
- Section 5.6, "Configuring a Form Fill Policy for Forms With Scripts," on page 153

5.1 Understanding an HTML Form

The following figure is an example of a Web page containing an HTML form:

Figure 5-1 *Sample HTML Form*

The information in this section uses this sample form to explain how to create a policy. This sample form deliberately contains a variety of field types:

- Input items for Username and Password
- Selection options for the Web server field
- Radio buttons for the role
- Check boxes for single sign-on

When you analyze a form, you need to decide if you want the policy to fill in all the fields or just some of them. You then need to look at the source HTML of the form to discover the names of the fields and their types.

An HTML form is created using a set of HTML tags. A form consists of elements such as fields, menus, check boxes, radio buttons, and push buttons that control how the form is completed and submitted. For more detailed information about forms, see the Forms section at www.w3.org (http://www.w3.org/TR/html401/interact/forms.html).

The following HTML data corresponds to the sample form (see Figure 5-1). The lines that contain the information needed to create a Form Fill policy appear in bold type. Each line corresponds to a field in the form that requires information or allows the user to select information.

In the example, each bold line contains information about a field, its name, and type. You use this information in the policy to specify how the information in the field is filled.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Form Fill Test Page</title>
</head>
<body>
  <form name="mylogin" action="validatepassword.php" method="post"
        id="mylogin">
    <table align="center" border="0" cellpadding="4" cellspacing="4">
      <tr align="center" valign="top">
        <td>
          <p align="center"><font size="5">Novell Services Login
             </font></p>
          <table align="center" border="0">

            <tr align="left">
              <td>Username:</td>
              <td><input type="text" name="username" size="30"></td>
            </tr>

            <tr align="left">
              <td>Password:</td>
              <td><input type="password" name="password" size="30">
              </td>
            </tr>

            <tr align="left">
              <td>City of<br>Employment:</td>
              <td><input type="text" name="city" size="30"></td>
            </tr>

            <tr align="left">
              <td>Web server:</td>
              <td>
                <select name="webserv" size="1">
                  <option value="default" selected>
                    --- Choose a server ---
```

```
                  </option>
                  <option value="Human Resources">
                    Human Resources
                  </option>
                  <option value="Development">
                    Development
                  </option>
                  <option value="Accounting">
                    Accounting
                  </option>
                  <option value="Sales">
                    Sales
                  </option>
                </select>
              </td>
            </tr>

            <tr>
              <td colspan="2" align="left" height="25" valign="top">
                <p></p>
              </td>
            </tr>

            <tr align="left">
              <td>Please specify<br>your role:</td>
              <td>
                 <input name="role" value="admin" type="radio">
                        Admin<br>
                 <input name="role" value="engineer" type="radio">
                       Engineer<br>
                 <input name="role" value="manager" type="radio">
                       Manager<br>
                 <input name="role" value="guest" type="radio">Guest
              </td>
            </tr>

            <tr>
              <td colspan="2" align="left" height="25" valign="top"
                  width="121">
                <p></p>
              </td>
            </tr>

            <tr align="left">
              <td>Single Sign-on<br>to the following:</td>
              <td>
                <input name="mail" type="checkbox">Mail<br>
                <input name="payroll" type="checkbox">Payroll<br>
                <input name="selfservice" type="checkbox">
                             Self-service<br>
              </td>
            </tr>
          </table>
        </td>
      </tr>

      <tr>
        <td colspan="2" align="center">
          <input value="Login" type="submit">
          <input type="reset">
        </td>
      </tr>
    </table>
  </form>
</body>
</html>
```

5.2 Creating a Form Fill Policy for the Sample Form

The sample form has ten input fields and five selection options that need to be configured in the Form Fill policy. The following steps explain how to create a shared secret to store the values and use that shared secret to create a Form Fill policy for this sample form. For information on configuring the form fill policy for a complicated form with JavaScript, see Section 5.6, "Configuring a Form Fill Policy for Forms With Scripts," on page 153.

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New*.
3. Specify a display name for the policy and select *Access Gateway: Form Fill* for its type.

4. (Optional) Specify a description for the Form Fill policy. This is useful if you plan to create multiple Form Fill policies.

   You might want to specify the name of the HTML page that contains the form this policy is designed to fill.
5. In the *Actions* section, click *New,* then select *Form Fill*.

6 In the *Form Selection* section, select *Form Name* and specify *mylogin* in the text box. The form name comes from the HTML page. See the following line in the source for the page:

```
<form name="mylogin" action="validatepassword.php" method="post"
        id="mylogin">
```

7 In the *Fill Options* section, specify all the input fields and select options. For each new field, click *New*. Specify the fields in the order in which they appear on the form. For items that are not available in the other data types such as an LDAP or Liberty attribute, create shared secrets to store the value.

The following table displays the Fill Options selected for each input field:

Form Name	Fill Options
username	**Input Field Name:** username
	Input Field Type: Text
	Input Field Value: Credential Profile: LDAP Credentials: LDAP User Name
password	**Input Field Name:** password
	Input Field Type: Password
	Input Field Value: Credential Profile: LDAP Credentials: LDAP Password

Form Name	Fill Options
webserv	**Input Field Name:** webserv **Input Field Type:** Select **Input Field Value:** Shared Secret: sampleLogin: webserv To create this shared secret, click *New Shared Secret*, specify *sampleLogin*, then click *OK*. Select *sampleLogin*, click *New Shared Secret Entry*, specify webserv, then click *OK*. For more information, see Section 5.4, "Creating and Managing Shared Secrets," on page 150. To add more entries to the same secret store, such as role and mail, you need to manage the secrets from the Identity Server. Save your draft of the policy, then click *Devices > Identity Servers > Shared Settings > Custom Attributes*. Select the name of your secret store (in this example it is sampleLogin). Add the entries you need for role, mail, payroll, and selfservice. These names need to match the form name.
role	**Input Field Name:** role **Input Field Type:** Radio Button **Input Field Value:** Shared Secret: sampleLogin: role
mail	**Input Field Name:** mail **Input Field Type:** Checkbox **Input Field Value:** Shared Secret: sampleLogin: mail
payroll	**Input Field Name:** payroll **Input Field Type:** Checkbox **Input Field Value:** Shared Secret: sampleLogin: payroll
selfservice	**Input Field Name:** selfservice **Input Field Type:** Checkbox **Input Field Value:** Shared Secret: sampleLogin: selfservice

8 In the *Submit Options* section, fill in the following fields:

Auto Submit: Select this option to submit the form as soon as all the values are filled in. If this option is not selected, even though all the values are filled in for the user, the user must click the *Submit* button.

Debug Mode: Select the *Debug Mode* option, which allows you to verify that the information is correct before submitting the form. If values must be filled in, you first see the form to add the values. When the form is submitted, you are presented with a JavaScript that contains all of the name/value pairs. To submit the form, you need to click the *Submit* button.

Insert Text in Header: Select this option so you can add a static value. In the *Text to Insert* box, specify the city value.

```
city = Provo
```

9 To create a login failure policy, click *New* in the *Actions* section, then select *Form Login Failure*.

10 In the *Form Selection* section, select *Form Name* and specify *mylogin* in the text box. The form name comes from the HTML page.

11 In the *Login Failure Processing* section, fill in the following field:

Clear Shared Secret Data Values from Policy: Select this option to clear the data stored in the Shared Secret object when login fails. Select the name you have given to this policy.

12 Use the up-arrow button to move the Form Login Failure policy to the top of the policy list.

You want the failure policy to execute first on login failure.

13 Click *OK*.

14 On the Policies page, click *Apply Changes*.

5.3 Implementing Form Fill Policies

Section 5.2, "Creating a Form Fill Policy for the Sample Form," on page 134 section describes how to create a simple Form Fill policy for a few input fields. This section describes all available options and explains how to use them to create a Form Fill policy and a Login Failure policy.

- Section 5.3.1, "Designing a Form Fill Policy," on page 137
- Section 5.3.2, "Creating a Form Fill Policy," on page 142
- Section 5.3.3, "Creating a Login Failure Policy," on page 147
- Section 5.3.4, "Troubleshooting a Form Fill Policy," on page 148

5.3.1 Designing a Form Fill Policy

Besides analyzing the form and determining the data items that need to be filled (see Section 5.1, "Understanding an HTML Form," on page 131), you need to consider the following when designing the Form Fill policy:

- "Verifying the Content or Page Type of the Form" on page 138
- "Creating a Form Matching Rule" on page 138
- "Including JavaScript in a Form Fill Policy" on page 140
- "Form Fill Character Sets (UTF-8)" on page 142

Verifying the Content or Page Type of the Form

When configuring the protected resource that uses a Form Fill policy, the URL in the *URL Path List* should include the filename of the page that contains the form. Sometimes this is not possible. If the URL references a directory, the Access Gateway has to parse the files that match the URL and determine which one contains the form.

The Access Gateway Appliance checks the files for the following content types:

```
text/html
text/xml
text/css
text/javascript
application/javascript
application/x-javascript
```

If a file has no content type or has a type other than one in the above list, the Access Gateway Appliance skips the file.

The Access Gateway Service does not check for content type; it just parses the files that match the URL.

Creating a Form Matching Rule

To create a successful Form Fill policy, you need to create a matching rule that matches the policy to the HTML page that contains the form, and then matches the form on the page. The Access Gateway uses the following rules, in the order listed, when determining whether a page contains the required form:

1. Matches the protected resource path in the URL with the page. If they don't match, the page is rejected. If they match, continues. For more information, see "Using the URL of the Protected Resource" on page 139.
2. Checks for CGI criteria. If they don't match, the page is rejected. If they match or no criteria is specified, continues. For more information, see "Using CGI Matching Criteria" on page 139.
3. Checks for page matching criteria. If they don't match, the page is rejected. If they match or no page matching criteria is specified, continues. For more information, see "Using Page Matching Criteria" on page 139.
4. Checks the form name criteria (which can be the `<FORM>` name attribute, the `<FORM>` ID attribute, or a number). If it doesn't match, the page is rejected. If it matches, the form is processed. For more information, see "Using Form Name Criteria" on page 140.

When the Access Gateway uses URL or CGI criteria, it can make a match early in the filling process. This allows the Access Gateway to fill the data from the Web server and send it, almost simultaneously, to the browser. However, if the Access Gateway is configured to use page matching criteria, the Access Gateway must retrieve the entire page from the Web server, process it, and then determine whether the page needs to fill a form. All this processing must be completed before the Access Gateway can send any data to the browser. Unless the page is quite small, users will clearly perceive the delay.

The form name matching criteria are not used for page matching. They are used to determine which form on the page is selected. Use the following methods to match the page and the form:

- "Using the URL of the Protected Resource" on page 139
- "Using CGI Matching Criteria" on page 139
- "Using Page Matching Criteria" on page 139
- "Using Form Name Criteria" on page 140

Using the URL of the Protected Resource

When you assign a Form Fill policy to a protected resource, we recommend that the URL specified in the *URL Path List* contain the filename of the page. Usually, such a URL is enough to match the HTML page for the form. However, when pages are dynamically generated, the same filename is sometimes used to display different pages. Sometimes you can't specify the filename in the URL. When this is the case, you need to use either the *CGI Matching Criteria* or the *Page Matching Criteria* to create an accurate page matching rule.

Using CGI Matching Criteria

If the page for the URL changes with the CGI portion of the URL (the portion that follows the question mark (?) and also called the query string), you can enter the CGI value. For example, consider the following URL:

```
http://webaccess.novell.com/servlet/webacc?Action=User.logout
```

If this is your URL, you can enter `Action=User.logout` as the value in the text box for the *CGI Matching Criteria* option. If the page generated from this URL always contains the page you want to match, you do not need to add any additional page matching criteria.

Using Page Matching Criteria

If your URL of your protected resource has the following characteristics, you need to use page matching criteria:

- The URL does not contain any CGI data.
- The URL displays generated pages that vary in content. For example, if your form fill login page and the login failure page share the same URL, you need to use page matching criteria.

Page matching criteria are the most processing-intense form of matching and should be avoided if possible, but sometimes they are the only method available to identify the page with the correct form. For example, suppose you have a login failure page and login page that use the same URL, with no CGI data. You can use page matching criteria to ensure that the Access Gateway matches the Form Fill policies for login and for login failure to the correct pages. You need to examine the source code for each page, and identify a string at the top of the page that uniquely identifies the page.

For example, the login page might contain a `<TITLE>` element that names the application the user is logging in to. If the login failure page does not contain the same `<TITLE>` element, you can use the `<TITLE>` element to identity the login page. Suppose that this is true and the login page contains the following string:

```
<TITLE>Novell WebAccess</TITLE>
```

You would add this string as the value in the text box for the *Page Matching Criteria* option. Remember that white space is significant when white space is entered to the left of the value in the text box. To have the Access Gateway ignore white space, left-justify the value in the text box, or to ensure the correct amount of white space, copy and paste the HTML text directly from the source code of the Web page.

Now you need to uniquely identify the login failure page. If this page does not have a `<TITLE>` element, look at the strings near the top of the page. Suppose the page contains the following string:

```
"Please log in again. You might have typed your name or password incorrectly."
```

Because the login page does not contain this string, you can use this string to identify the login failure page. You would add the following string as the value in the text box for the *Page Matching Criteria* option for the login failure Form Fill policy.

```
Please log in again.
```

To have the Access Gateway ignore white space, left-justify the value in the text box, or to ensure the correct amount of white space, copy and paste the HTML text directly from the source code of the Web page.

Using Form Name Criteria

After identifying the page, the Access Gateway needs to identify the form on the page. If there is only one form on the HTML page, the Access Gateway can easily identify the form. If the form has a name or an ID attribute, you can use the value of the attribute to identify the form. If the form doesn't have either of these attributes, you can use the *Number* option with a value of 1. The first form the Access Gateway finds on the page matches.

When multiple forms exist on the same HTML page, the easiest and fastest matching method is to give each form a unique name or unique ID on the HTML page. If the forms have the same name or ID, you need to use the Number option, and the order in which they appear on the page determines their number.

The value 0 for the *Number* option has special meaning. You use this value when you want the Form Fill policy to fill in values for all forms on the page. Sometimes a page has multiple forms, but all forms on the page must be filled in before the page can be submitted. For example, one form might contains user information and another form contain user preferences. If both of these forms need to be filled in before the user can log in, then you can use the Number option set to 0, and the Fill Options section of the policy can contain fields for both forms, in the order in which they appear on the page.

Including JavaScript in a Form Fill Policy

The following figure illustrates a simple form:

Figure 5-2 *Form Login Page*

The source code for this simple form reveals that it includes JavaScript functions:

```
<html><head><title>Login Page</title></head><body>
<h1 align="center">Login Page</h1>
<script language="JavaScript">
  function setCookie(){
    document.cookie="myCookieName=myCookieValue";
  }
  function validate(){
  if(document.mylogin.title.ldap.length == 0){
    alert("You must provide the title for the user!");
    return false;
  }
  return true;
}
</script>
<form name="jscript" action="viewInfo.php" method="post" onload="setCookie()">
<center>
<table border="1" cellpadding="4" cellspacing="4">
  <tbody><tr>
  <td>Username:</td>
  <td><input name="username" size="30" type="text"></td>
  </tr>

  <tr>
  <td>Title:</td>
   <td><input name="title" size="30" type="text"></td>
  </tr>
  <tr>
   <td>Password:</td>
  <td><input name="password" size="30" type="text"></td>
  </tr>

  <tr>
    <td>LDAP SERVER:</td>
  <td><input name="ldap" size="30" type="text"></td>
  </tr>
  <tr>
  <td colspan="2" align="center">
  <input value="Login" onclick="return validate();" type="submit">
  </td>

  </tr>
</tbody></table>
</center>
</form>

<script language="JavaScript">
function doCookie(){
document.cookie="myCookieName=myCookieValue";
}
return true;
}
</script>

</body></html>
```

The significant code snippets for determining whether to include JavaScript commands in the Form Fill policy are displayed in bold. The `<script>` elements are in bold because you need to be aware of all the JavaScript on the HTML page. Whether all the functions in the JavaScript need to be included in the policy is usually determined by trial and error. There are some clues you can use to determine the requirements:

- If a function is called within the form, you should include it in the Form Fill policy. The above form calls two JavaScript functions, `setCookie()` and `validate()`.
- If a function is not called by the form, you probably do not need to include it. The above form has one JavaScript function that falls within this category, `doCookie`. You can probably leave out these types of functions, but only trial and error can determine whether that is true.

For this form, select the *Auto Submit* option and the *Enable JavaScript Handling* option. If you wanted to test whether the `doCookie()` function was needed, you would specify the following in the *Functions to Keep* text box:

```
function setCookie()
function validate()
```

Each function needs to be placed on a separate line. This feature does a string compare, so the string after the function key word must match exactly a string in the JavaScript.

Form Fill Character Sets (UTF-8)

Access Manager supports only UTF-8 encoding (UCS Transformation Format 8) and ISO 8859-1. Otherwise, Form Fill translations to the secret data store cannot be guaranteed.

5.3.2 Creating a Form Fill Policy

1. Examine the source code for the HTML form and determine what data the form requires and where that data is stored (LDAP attributes, Liberty User Profile attributes, shared secrets, credential profiles, etc.)

   Ideally, the form should be its own HTML page, and the page should be as small as possible. Form Fill must parse the entire file and assemble the body in contiguous memory before the first byte of the form is displayed to the user. For a large file, this can take enough time that your users might think the system has a problem.

   If it isn't possible to have the form on its own HTML page, make sure the form is easily identifiable on the page. For example, give the form a name or use CGI data (the text that the follows the question mark in the URL) to identify the page and form.
2. In the Administration Console, click *Policies* > *Policies*.
3. Select the policy container, then click *New*.
4. Specify a name for the policy, select *Access Gateway: Form Fill* as its *Type*, then click *OK*.
5. Fill in the following fields:

   Description: (Optional) Describe the purpose of this policy. Because Form Fill policies are customized to match the content of a specific HTML page, you might want to include the name of the page as part of the description.

   Priority: Determines the order in which a rule is applied in the policy, when the policy has multiple rules. Form Fill does not use this field.
6. In the *Actions* section, click *New* and select *Form Fill*.

7 In the *Form Selection* section, specify how the Access Gateway can identify the form on the page. Select one or more of the following methods. Be specific and use as few of the methods as possible. For information on how to use these options effectively, see "Creating a Form Matching Rule" on page 138.

Form Name: Identities the form on the HTML page. Select one of the following:

- **Form Name:** If the `<form>` element on your HTML page specifies a name attribute, select *Form Name* and specify the value of the name attribute in the text box. For example, suppose your form contains the following:

  ```
  <form name="mylogin" action="validatepassword.php" method="post"
  id="form1">
  ```

  For this form, you would specify *mylogin* in the text box.

- **Form Number:** The Access Gateway numbers forms sequentially from the top of the HTML page. If your page has multiple forms, you can use *Form Number* option and specify the form's sequential location in the text box.

- **Form ID:** If the `<form>` element on your HTML page specifies an id attribute, select *Form ID* and specify the value of the id attribute in the text box.

  For example, if your form contains the following:

  ```
  <form name="mylogin" action="validatepassword.php" method="post"
  id="form1">
  ```

  For this form, you would specify *form1* in the text box.

For more information, see "Using Form Name Criteria" on page 140.

CGI Matching Criteria: Allows the Access Gateway to evaluate the query string in the URL (the portion after the question mark) to differentiate pages that have the same URL. Consider the following URL:

```
http://webaccess.novell.com/servlet/webacc?Action=User.login
```

For this URL, enter the following string in the text box for *CGI Matching Criteria*:

```
Action=User.login
```

If possible, copy the text from the form and paste it into the *CGI Matching Criteria* text box.

For more information, see "Using CGI Matching Criteria" on page 139.

Page Matching Criteria: Causes the Access Gateway to search the HTML page for the specified text. If the specified text is found on the page, the page is a match for the policy. If it isn't found, the page is not a match for the policy and the policy is not applied. For example, suppose your HTML page has the following string within the `<FORM>` element:

```
<title>Form Fill Test Page</title>
```

If you enter this string in the *Page Matching Criteria* box, the Access Gateway searches the form for this string. If it finds the string, it knows it has a match.

White space is significant. If the text in the text box is left-justified, the text can be found anywhere on the HTML page. If the text contains leading white space, such as ten spaces, the text must be found with ten leading spaces. If possible, copy the text as it appears on the form and paste it into *Page Matching Criteria* text box.

The more specific your information is, the faster Access Gateway can match the form. Parsing page matching criteria is a very intensive process. If possible, use the URL path specified for the protected resource or *CGI Matching Criteria* to identity the form.

For more information, see "Using Page Matching Criteria" on page 139.

8 In the *Fill Options* section, create an entry for all the input fields and select options in the form. For each input field or select option, you need to specify the following information:

Input Field Name: Specifies the name of the field or option. This is the name attribute of the element on the form.

Input Field Type: Specifies the type attribute for the input field or select option in the form. Select one of the following data types for the field:

- **Text:** Indicates that the field is a text field on the form.
- **Password:** Indicates that the field is a password field on the form.
- **Checkbox:** Indicates that the field is a check box on the form.
- **Radio Button:** Indicates that the field is a radio button on the form.
- **Select:** Indicates that the field is a select option on the form.
- **Hidden:** Indicates that the field is an input field, but that this field is hidden from the user.
- **Not Specified:** Indicates that the field is an input field, but the data type is not specified in the form.

Input Field Value: Specify the value for the field. You must specify the data type, then enter the value. Select one of the following data types:

- **Credential Profile:** Specifies that the value should be retrieved from the credentials the user specified during authentication. If you have created a custom contract that uses credentials other than the ones listed below, do not use the Credential Profile as an input value.
  - **LDAP Credentials:** If you prompt the user for a username and password, select this option, then either *LDAP User Name* (the cn of the user) or *LDAP User DN* (the fully distinguished name of the user). Your Web server requirements determine which one you use.

    The default contracts assign the cn attribute to the Credential Profile. If your user store is an Active Directory server, the SAMAccountName attribute is used for the username and stored in the cn field of the LDAP Credential Profile.

- **X509 Credentials:** If you prompt the user for a certificate, select this option, then select one of the following option depending on your Web server requirements.
  - **X509 Public Certificate Subject:** Specifies that the subject field from the certificate should be the value, which can match the DN of the user, depending upon who issued the certificate.
  - **X509 Public Certificate Issuer:** Specifies that the issuer field from the certificate should be the value, which is the name of the certificate authority (CA) that issued the certificate.
  - **X509 Public Certificate:** Specifies that the entire certificate should be the value.
  - **X509 Serial Number:** Specifies that the certificate serial number should be the value.
- **SAML Credential:** Injects the SAML assertion as the value of the field when SAML is used for authentication. This value is usually used for the user's password.

- **LDAP Attribute:** Indicates that the value should be retrieved from the specified LDAP attribute. If the attribute you require does not appear in the list, click *New LDAP Attribute* to add the attribute.

  The *Refresh Data Every* option allows you to determine when to send a query to the LDAP server to verify the current value of the attribute. Because querying the LDAP server slows down the processing of a policy, LDAP attribute values are normally cached for the user session.

  Change the value of this option from session to a more frequent interval only on those attributes that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes.
- **Liberty User Profile:** Indicates that the input field contains a Liberty User Profile attribute. In the value field, select the attribute. The attribute you select must be mapped to an LDAP attribute, and the Access Gateway retrieves its value from the LDAP directory.
- **Shared Secret:** Indicates that the input field contains a user-entered value that is to be stored in the specified shared secret store.

  You can create your own value. Click *New Shared Secret*, specify a display name for the store, and Access Manager creates the store. Select the store, click *New Shared Secret Entry*, specify a name for the attribute, then click *OK*. The store can contain one name/value pair or a collection of name/value pairs. For more information, see Section 5.4, "Creating and Managing Shared Secrets," on page 150.

  The *Refresh Data Every* option allows you to determine when to send a query to verify the current value of the secret. Because querying slows down the processing of a policy, secret values are normally cached for the user session.

  Change the value of this option from session to a more frequent interval only on those secrets that are critical to the security of your system or to the design of your work flow. You can select to cache the value for the session, for the request, or for a time interval varying from 5 seconds to 60 minutes.
- **String Constant:** Indicates that the input field contains a static value. In the text box, specify the value for the string constant.
- **Data Extension:** (Conditional) If you have installed a data extension for Form Fill policies, injects the value that the extension retrieves. For more information about creating a data extension, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Nacm).

NOTE: To improve the policy's performance, configure the LDAP Attributes, Credential Profile, Liberty User Profile, and Shared Secret attributes to be sent with authentication. For more information, see "Configuring the Attributes Sent with Authentication"in the *NetIQ Access Manager 3.2 Identity Server Guide*.

Data Conversion: Specify whether the case of the value entered by the user should be converted. Select one of the following options:

- **None:** Indicates that no conversion should be performed on the value.
- **To Upper Case:** Indicates that the value should be converted to uppercase.
- **To Lower Case:** Indicates that the value should be converted to lowercase.
- **LDAP DN to NDAP Partial Dot Notation:** Converts the LDAP DN (which uses typed comma notation) to eDirectory™ typeless dot notation.

  `cn=jsmith,ou=Sales,o=novell to jsmith.sales.novell`

- **LDAP DN to NDAP Leading Partial Dot Notation:** Converts the LDAP DN to eDirectory typeless leading dot notation.

  `cn=jsmith,ou=Sales,o=novell to .jsmith.sales.novell`

- **LDAP DN to NDAP Fully Qualified Partial Dot Notation:** Converts the LDAP DN to eDirectory typed dot notation.

  `cn=jsmith,ou=Sales,o=novell to cn=jsmith.ou=Sales.o=novell`

- **NDAP Fully Qualified Leading Dot Notation:** Indicates eDirectory typed leading dot notation.

  `.cn=jsmith.ou=Sales.o=novell`

9 In the *Submit Options* section, specify how you want the information in the form submitted to the Web server. (The HTML form page determines whether the post method or the get method is used for the submission.) Select one or more of the following options:

Auto Submit: Indicates that you want the form submitted to the Web server without having the user confirm the submission by clicking a *Submit* button. If this option is not selected, Form Fill can fill in the data, but the user must click the *Submit* button before the data is sent to the Web server. When the form is not auto submitted, all the JavaScript on the form is executed.

If you select *Auto Submit*, you can select one or more of the following options:

- **Debug Mode:** Allows you to verify that the information in the filled-in form is valid before it is posted to the Web server. You can right-click and view the source that is being submitted to the Web server. If it is correct, click *Submit* to send it to the Web server.

  This is a troubleshooting option. We recommend that you use it when creating a new Form Fill policy, and that you remove it when you have determined that the policy is behaving as expected.

- **Mask Data:** Replaces text input field values (username, password, etc.) with nov-ss-ff-masked instead of the value specified by the value parameter when the form is sent to the browser. The Access Gateway replaces these masked values with the real values when the Access Gateway submits the form to the Web server. The user's browser never sees the actual values for these fields.

Insert Text in Header: If this option is selected, you can use the *Text to Insert* option to specify text to add to the header. Use this option to insert static values into the form.

Enable JavaScript Handling: Retains JavaScript from the original page if you have also selected the *Auto Submit* option. For a new Form Fill policy, you should also select the *Debug Mode* option so you can verify that you have included all the functions and statements that need to be executed in the policy.

Use the following fields to specify how you want the JavaScript handled:

- **Functions to Keep:** Specifies the functions you want executed from the JavaScript on the original page. By default, no functions on the page are executed. In the text box, use the following format:

  ```
  function setCookie()
  ```

  where `function` is a key word, followed by a space, and then the name of the function. Each function should be entered on a separate line, but you need only one function per script block. Everything must match exactly (name, capitalization, white space.) If you include the parentheses after the function name (setCookie()), they must exactly match the white space in the JavaScript. If possible, copy the function from the HTML page.

- **Statements to Execute on Submit:** Specifies the functions you want executed just before the form is posted. Copy the JavaScript statement from the HTML page or add a JavaScript statement that you want called that is not on the HTML page. This allows you to modify the behavior of the form when you can't modify the form.

  If the text box is empty, the JavaScript function specified in the submit field of the HTML page executes before the form is posted.

For more information, see "Including JavaScript in a Form Fill Policy" on page 140.

10 In the *Error Handling* section, specify how you want errors handled.

   Redirect to URL: When an LDAP or NSS error occurs, the user is redirected to the URL you specify in the text box. This is optional and allows you to customize the error handling process. If you do not customize it, a standard error page is displayed.

11 Click *OK*, then click *Apply Changes*.

12 Continue with Section 5.3.3, "Creating a Login Failure Policy," on page 147 or "Assigning a Form Fill Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.

5.3.3 Creating a Login Failure Policy

The Login Failure policy can be part of the same policy as the Form Fill policy, if both share the same URL. In this case, the Form Login Failure policy should be the first action in the policy, and the Form Fill policy should be the second action in the policy. This causes a login failure to execute the policy that clears the stored data and the Form Fill policy to prompt the user for new data.

If the user is redirected to a different page when login fails, it is best to create a separate policy for that page, create a protected resource that includes just that page, and assign your Form Login Failure policy to that resource.

To create a Login Failure policy:

1 In the Administration Console, click *Policies* > *Policies*.

2 Select the policy container, then click *New*.

3 Specify a name for the policy, select *Access Gateway: Form Fill* as its *Type*, then click *OK*.

4 In the *Actions* section, click *New*, then select *Form Login Failure*.

5 In the *Form Selection* section, identify the form. This section uses the same criteria for identifying a form as the Form Fill policy. For more information, see Step 7 on page 143 and "Creating a Form Matching Rule" on page 138.

6 In the *Login Failure Processing* section, define the actions you want executed when a user fails to log in. Fill in the following fields:

Redirect to URL: When a user's login attempt fails, use this option with its text box to specify the URL you want the user redirected to. This is optional and allows you to customize what happens on login failures.

Clear Shared Secret Data Values From Policy: Select this field to delete the user's stored data for a Form Fill policy. If the user has the ability (and perhaps the requirement) to periodically change his or her password or any other information on the form, you need to select this field. Otherwise, the wrong data can be stored for the user, and the Access Gateway has no way of updating the information.

From the list of Form Fill policies, select the policy whose stored values should be cleared with this Login Failure policy.

7 Click *OK*, then click *Apply Changes*.

8 Continue with "Assigning a Form Fill Policy to a Protected Resource" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.

5.3.4 Troubleshooting a Form Fill Policy

When a new Form Fill policy is not behaving as expected, use the following tips to discover the cause:

- Select the *Debug Mode* option. This option prepares the form for submission, but doesn't submit the form until you click the *Submit* button. This allows you to view the source, and determine if the policy is generating the required data.

- Ensure that all input fields have valid names, that the fields are being filled in the correct order, and that any JavaScript commands have been entered correctly.
- Enable Form Fill logging. Form Fill is a function of both the proxy service and the Embedded Service Provider. The Embedded Service Provider logs the evaluation of the policy, and the proxy logs the process of gathering the data. To enable the Embedded Service Provider tracing, see Section 7.1, "Turning on Logging for Policy Evaluation," on page 161. To enable Access Gateway log entries for Form Fill policies, see "Enabling Form Fill Logging" on page 175.

Check for the following problems with the source content of the Form Fill page:

- "Valid HTML Structure" on page 149
- "The Option Element Does Not Contain a Value Attribute" on page 149
- "The Form Element Does Not Contain a Method Attribute" on page 150

Valid HTML Structure

The Form Fill process aborts if the page does not contain valid HTML structure. The page must contain the `<html></html>` tags, and the form must contain the `<form></form>` tags. If these tags are missing, you should correct the source page on the Web server. If this is not possible, you can create a rewriter policy to add the tags.

- To add the `<html>` tag, have the rewriter policy search for the `<body>` tag, and replace it with `<html><body>`.
- To add the `</html>` tag, have the rewriter policy search for the `</body>` tag, and replace it with `</body></html>`.
- Use similar entries to add the `<form></form>` tags. You'll need to discover which tag or phrase starts and stops the form.

Configure your rewriter policy so that it runs before the default rewriter policy. For more information about rewriter policies, see "Configuring HTML Rewriting" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.

The Option Element Does Not Contain a Value Attribute

If an `<option>` element does not contain a value attribute, Form Fill cannot fill the value. For example:

```
<form action="select.htm">
   <select name="top2">
      <option>Bob</option>
      <option>Alice</option>
   </select>
</form>
```

If your form contains `<option>` elements similar to these, they need to rewritten to contain a value attribute. For example:

```
<form action="select.htm">
   <select name="top2">
      <option value="name1">Bob</option>
      <option value="name2">Alice</option>
   </select>
</form>
```

If possible, change the source page on the Web server to add the value attribute to the `<option>` elements. If this is not possible, you can use a rewriter policy to add the value attribute.

- For the Bob option, have the rewriter policy search for `<option>Bob` and replace it with `<option value="name1">Bob`.
- For the Alice option, have the rewriter policy search for `<option>Alice` and replace it with `<option value="name1">Alice.`

Configure your rewriter policy so that it runs before the default rewriter policy. For more information about rewriter policies, see "Configuring HTML Rewriting" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.

The Form Element Does Not Contain a Method Attribute

If the `<form>` element does not contain a method attribute, Form Fill does not run an Auto Post. For example, the following form cannot use an Auto Post.

```
<form name="loginForm">
```

To enable Form Fill so that it can run an Auto Post, you need to add a method attribute to the `<form>` element. For example:

```
<form method="get" action="index.htm" name="loginForm">
```

If possible, change the source page on the Web server to add the method attribute to the `<form>` element. If this is not possible, you can use a rewriter policy to add the method attribute.

- Search for `<form`
- Replace this string with `<form method="get" action="index.htm"`

Configure your rewriter policy so that it runs before the default rewriter policy. For more information about rewriter policies, see "Configuring HTML Rewriting" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.

5.4 Creating and Managing Shared Secrets

A shared secret is an object that holds name and value pairs for Form Fill and Identity Injection policies.

- If your HTML form prompts the user for more than credential information, you need to create a shared secret to store the values.
- If your Web server requires some name/value pairs to be injected and these are not available from the HTTP request, you need to create a shared secret to store these name/value pairs so that they can be injected into the header before it is sent to the Web server.

Access Manager supports the creation and use of secrets from the following locations:

- In the local configuration store
- In eDirectory user stores that are running Novell SecretStore
- In a user store that has been configured with a custom attribute for secrets

For more information on configuring Access Manager to store secrets, see "Configuring a User Store for Secrets" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

This section describes the following topics:

- Section 5.4.1, "Naming Conventions for Shared Secrets," on page 151
- Section 5.4.2, "Creating a Shared Secret Independent of a Policy," on page 151
- Section 5.4.3, "Modifying and Deleting a Shared Secret," on page 152

5.4.1 Naming Conventions for Shared Secrets

The policy engine allows you to create shared secrets and name the attributes for the store as you are creating an Identity Injection or Form Fill policy. When you create the shared secret, we recommend that you name the shared secret after the application for which you are creating the policy. Each value requires a name, and we recommend that you use the same name for the value name as the Input Field Name on a Form Fill policy or for the header name on an Identity Injection policy. For example if your e-mail application requires the e-mail address for the name on the login form, you could set up the following Shared Secret values:

Input Field Name	Input Field Value	Shared Secret Name	Entry Name
emailaddress	Shared Secret	emailapp	emailaddress

Your applications, how you use them, and your personal preferences determine whether you create one shared secret and use it for all your applications or whether you create a shared secret for each application.

- If the applications use some of the same secrets, you can use the same shared secret for these applications. In this case, give the shared secret a name that reflects all of the applications using it.
- If an application does not use the same secrets as another application and you want the freedom to remove the application and its secrets without affecting other applications, you should create a separate shared secret for this application.
- If you are using Novell SecretStore, the secret names specified in your Access Manager policies need to match the names you have already configured.

A local shared secret store does not contain any name/value pairs until you configure a Form Fill policy to add name/value pairs or enable the *Allow End Users to See Credential Profile* option. This option allows the username and password to be stored in the local secret store. To set this option, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Providers* > *Credential Profile*.

5.4.2 Creating a Shared Secret Independent of a Policy

You can create a shared secret as part of the process of creating a Form Fill or Identity Injection policy. You can also create a shared secret independent of a policy:

1. In the Administration Console, click *Devices* > *Identity Servers*, then click *Shared Settings* > *Custom Attributes*.
2. To create a new shared secret, click *New* in the *Shared Secret Names* section, and fill in the following fields:

   Secret Name: Specify a display name for the shared secret.

   Secret Entry Name. Specify an attribute name for a value you want to store.
3. Click *OK*.

   The Identity Server creates and encrypts the object.

4 To create additional attributes to store values, click the secret name, click *New*, specify a name, then click *OK*.

5 Click *OK*.

5.4.3 Modifying and Deleting a Shared Secret

Before deleting a shared secret, you need to delete the policies that are using the shared secret or modify the policies to use a different shared secret. For information about deleting policies, see Section 1.3.3, "Deleting Policies," on page 11.

Both Form Fill and Identity Injection policies can use shared secrets. The following instructions explain how to modify an Identity Injection policy to use a new shared secret and then how to delete the old shared secret.

1 In the Administration Console, click *Policies* > *Policies* > *[Name of Policy]* > *[Rule]*.

2 Select the *Value* field that uses the shared secret you want to delete. Click its name, then click *New Shared Secret*.

3 Specify the name for a new shared secret, then click *OK*.

4 Click the name of the shared secret, select the new shared secret store, then click *New Shared Secret Entry*.

5 Specify the attribute name for this shared secret entry, then click *OK*.

6 Modify any other *Value* fields to use the new shared secret. Create new attributes as needed.

7 To save the modifications to the policy, click *OK* twice, then *Apply Changes*.

8 To delete the old shared secret, click *Identity Servers* > *Shared Settings* > *Custom Attributes*.

9 Select the name of the old shared secret and the attributes, then click *Delete*.

5.5 Importing and Exporting Form Fill Policies

You can import and export the Form Fill policies in order to run them in other Access Manager configurations and to analyze the policy. The policy is exported as a text file with XML tags. We do not recommend editing the exported file with a text editor. Any changes you want to make to a policy ought to be done through the Administration Console.

To export a Form Fill policy:

1 In the Administration Console, click *Policies* > *Policies*.

2 Select a Form Fill policy, then click *Export*.

3 (Optional) Modify the name suggested for the file.

4 Click *OK*.

5 Using the features of your browser, specify where the file is be copied.

To import a policy:

1 Make sure any referenced shared secret stores have been created. See Section 5.4, "Creating and Managing Shared Secrets," on page 150.

2 If the policy uses LDAP or Liberty Profile attributes, make sure the Identity Server has been configured for these same attributes.

3 In the Administration Console, click *Policies* > *Policies*.

4 Click *Import*, then browse to the location of the file.

5 Click *OK*.

6 When the policy appears in the list, click *Apply Changes*.

5.6 Configuring a Form Fill Policy for Forms With Scripts

The Form Fill policy created for the Linux Access Gateway works well with forms that contain a *Submit* button whose `onclick` action submits the form data to the Web server without executing any JavaScript or VBScript. However, when HTML forms contain complicated JavaScript or VBScript, Form Fill for that form fails.

For example, single sign-on by using the Form Fill policy to fill and autosubmit a form fails if the Submit button or the login button requires execution of a JavaScript function before submitting the form data to the Web server.

The following sections explain why Form Fill fails with the Form Fill policy when the HTML form contains complicated JavaScript. This section also describes the procedure to configure a Form Fill policy for such forms.

- Section 5.6.1, "Why Does Form Fill Fail with the Default Policy?," on page 153
- Section 5.6.2, "Understanding How a Form Is Submitted," on page 155
- Section 5.6.3, "Creating a Form Fill Policy for Autosubmission," on page 156
- Section 5.6.4, "Configuring the Advanced Options for Autosubmission," on page 157

5.6.1 Why Does Form Fill Fail with the Default Policy?

The following section explains the process that takes place when a client requests a form that is configured with the Form Fill policy as described in Chapter 5, "Creating Form Fill Policies," on page 131.

Figure 5-3 *Sample Login Form with JavaScript*

When the Access Gateway is configured with the default Form Fill policy, it adds the following function to the Login page received from the Web server. The bold text indicates where JavaScript is called.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--Generated by Apache Software Foundation (Xalan XSLTC)-->
<html class="detail/detail">

<head>
     <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
          <script type="text/JavaScript">
             /*SET up global vars*/
            //all the variable declaration

          <script type="text/JavaScript">
             function dvdRegisterSelect() {
                    }
          </script>

     <title>Login Page</title>

</head>

<body id="tpz_body" style="width:99%; "onload="tpzOnLoad('login.prompt.g');
window.status='login.prompt.g'; ContextMenu.setup({'showForms':true});
ContextMenu.attach('detail/detail', 'cwc_optionsMenu_detail')"
onfocus="window.status='login.prompt.g'" >

       <script type="text/JavaScript">
          var arReenable = new Array();
          function enableAll() {return reenableControls(arReenable);}
       </script>

      <script type="text/JavaScript">
         //all the variable declaration
      function verify( f, bSubmitToSelf ){ return verifyFields
(bSubmitToSelf,"\n");}
     </script>

   <div>
      <a title="Login" class="tabSelected">Login</a>
   </div>
 <formname="topaz"id="topaz"method="post"action="detail.do"
onsubmit="enableAll();return verify(this,true);">
<input type="hidden" name="focus" id="focus" value="var/user.id">
<input type="hidden" name="focusContents" id="focusContents" VALUE="testuser1" >
<input type="hidden" name="focusId" id="focusId" VALUE="X2" >
<input type="hidden" name="formname" id="formname" VALUE="login.prompt.g">
<input type="hidden" id="clientWidth" name="clientWidth" VALUE="1473" >

     <script type="text/javascript">
        function printThisView(){tpzPrintDetail();}
     </script>

<input type="text" id="X2" name="var/user.id" dvdVar=""
onclick="handleOnClick(this,event);"  VALUE="testuser1" scripttype="text">
<input type="password" id="X5" name="var/old.password" dvdVar=""
onclick="handleOnClick(this,
event);" " VALUE="novell081" >

<input type="button" name="0" id="X8" ButtonID="0" title="Login Page" value="Login"
onclick="tpzDrillTable('', 'Login', '0','listdetail')" >
<input type="button" name="3" id="X9" ButtonID="3" title="Exit Login Page"
```

```
value="Cancel" onclick="tpzDrillTable('', 'Cancel', '3','listdetail')" >
</form>

       <script language="JavaScript">
          <!--
             function LAGSubmitForm()
                   {
            document.forms[0].submit();
                   }
            LAGSubmitForm();
          //-->
       </script>
</body>
</html>
```

In the above code, the `LAGSubmitForm()` function calls the default submit action of the form, which uses a POST request to send the data to the Web server. But the `submit` action for the sample login form requires a JavaScript function to be executed. This function in turn submits the form data to the Web server. However, because the JavaScript is not executed by the default Form Fill policy, posting of the form data fails:

```
row=&__x=&thread=0&event=0&transaction=0&type=detail&focus=var%2Fuser.id&focusCont
ents=testuser1&focusId=X2&focusReadOnly=&start=&count=&more=&tablename=&window=&cl
ose=&_blankFields=&_uncheckedBoxes=&formchanged=&formname=login.prompt.g&_multiSel
ection=&_multiSelection_tableId=&clientWidth=1473&var%2Fuser.id=testuser1&var%2Fol
d.password=novell081&var%2FL.language=en&0=Login&3=Cancel
```

Meanwhile, the browser expects to receive the following POST request and does not autosubmit the form:

```
row=&__x=&thread=0&event=0&transaction=0&type=detail&focus=var%2Fuser.id&focusCont
ents=testuser1&focusId=X2&focusReadOnly=null&start=&count=&more=&tablename=&window
=&close=&_blankFields=&_uncheckedBoxes=&formchanged=&formname=login.prompt.g&_mult
iSelection=&_multiSelection_tableId=&clientWidth=1217&var%2Fuser.id=testuser1&var%
2Fold.password=novell081&var%2FL.language=en
```

Note the difference in POST requests sent to the browser. The first POST request has `&0=Login&3=Cancel` appended, which causes the login to fail.

For the browser to send the proper POST data, the Linux Access Gateway must add the following JavaScript statement to the *Statements to execute* section:

```
tpzDrillTable('','Login','0','listdetail');
```

5.6.2 Understanding How a Form Is Submitted

For the Access Gateway Appliance, you can configure the Form Fill policy to submit the form in the following ways:

- **Manual Submit:** When a form is configured for manual submission, all the fields configured in the Form Fill policy are automatically filled by the Linux Access Gateway for the user. The user must then manually click the *Submit* button in the form to submit the form to the Web server protected by Linux Access Gateway.
- **Autosubmit:** When Autosubmit is configured, the actual form is processed in such a way that all additional scripts not required to submit the form data to the Web server are removed. A temporary form is created on runtime with necessary form data in hidden format and with an additional `LAGSubmitForm()` function as follows:

  ```
  function LAGSubmitForm()
  {
  executeJavaScript();
  }
  LAGSubmitForm();
  ```

In this example, `executeJavaScript()` is the function that executes the JavaScript or the VBScript statements configured in the *Statements to execute* section. If statements to be executed are present, you can also find the function definition for `executeJavaScript()` as follows:

```
executeJavaScript()
{
document.forms[0].submit();
}
```

In this example, `form[0]` is the single form in the HTML page and `submit` is the default action associated with the submit or login button of the form that automatically submits the form to the Web server. This approach works for forms where the default action of the *Submit* button is to submit a POST request for the form data.

- **Autosubmit with Masking:** When Autosubmit with masking is enabled for a form, the form data is submitted automatically to the Web server, but the data sent to the Web browser over the network is masked for additional security.
- **Submitting with the help of touch files:** If your form requires the execution of JavaScript when the form is submitted, you cannot use the Autosubmit options.This also means that single sign-on is disabled.

  To create a policy that allows autosubmitting for this type of form, you need to create the policy as described in Section 5.6.3, "Creating a Form Fill Policy for Autosubmission," on page 156 and create two touch files as described in Section 5.6.4, "Configuring the Advanced Options for Autosubmission," on page 157.

5.6.3 Creating a Form Fill Policy for Autosubmission

1. In the Administration Console, click *Policies* > *Policies*.
2. Select the policy container, then click *New*.
3. Specify a display name for the policy and select *Access Gateway: Form Fill* for its type.
4. (Optional) Specify a description for the Form Fill policy.
5. In the *Actions* section, click *New,* then select *Form Fill*.
6. In the *Form Selection* section, select *Form Name* and specify *topaz* in the text box.
7. In the *Fill Options* section, specify all the input fields and select the options that you want.
8. In the *Submit Options* section, select *Auto submit*.
9. Select *Enable JavaScript Handling*.

10. Select *Functions to Keep*, then specify the JavaScript functions that need to be retained when the form is being automatically submitted. For the example form, specify the following functions:

```
function dvdRegisterSelect()
function enableAll()
function verify(f, bSubmitToSelf)
function printThisView()
function tpzDrillTable(a,b,c,d)()
```

11 Click *OK*.

12 Select *Statements to Execute* and specify the form action that needs to be performed when the form is submitted. For the sample form, specify the following statement:

```
function executeJavaScript()
{
    tpzDrillTable('','Login','0','listdetail');
}
executeJavaScript();
```

You must perform this step in order to execute the functions configured in the *Functions to keep section* because the Linux Access Gateway does not process HTML to include the `LAGSubmitForm()` function.

13 Click *OK*.

14 On the Policies page, click *Apply Changes*.

5.6.4 Configuring the Advanced Options for Autosubmission

When HTML forms contain complex JavaScript or VBScript, you must enable the following two advanced options:

- **#NAGGlobalOptions InPlaceSilent=on:** This enables single sign-on to certain Web sites that require the login page to remain as is without any modifications to its structure. This option is equivalent to `.enableInPlaceSilentFill` in the 3.1 SP4 Access Gateway Appliance.
- **#NAGGlobalOptions InPlaceSilentPolicyDoesSubmit=on:** This option enables form fill in HTML pages with complex JavaScript or VBScripts. This option is equivalent to `.enableInPlaceSilentFillNew` in the 3.1 SP4 Access Gateway Appliance.

To enable these options:

1. In the *Administration Console*, click *Access Gateways* > *Edit* > *Advanced Options*.
2. Add the following in the *Advanced Options* list:

   ```
   NAGGlobalOptions InPlaceSilent=on
   NAGGlobalOptions InPlaceSilentPolicyDoesSubmit=on
   ```

3. Click *OK*.

6 Creating External Attribute Source Policies

Access Manager can be used as an identity provider for several different third-party service providers. Some of these service providers require attributes that are not part of the user store where the user is authenticated. The External Attribute Source policy allows you to retrieve attributes from the external sources.

You can configure this policy with rules to retrieve the attributes. A rule can contain all the conditions available for a policy. To enable this policy, you need to write a data extension class or provide a string constant value. The data extension class can be configured with constant and dynamic properties like any other policies.

For information about the structure and template of a data extension class and example code, see the Policy Extension API in the Access Manager 3.2 Developer Kit at NetIQ Access Manager Developer Tools and Examples (http://developer.novell.com/documentation/nacm32/nacm_enu/data/bookinfo.html).

6.1 Configuring an External Attribute Source Policy

1. In the Administration Console, click *Policies* > *Policies*.
2. Click *New*.
3. Specify a name for the policy, select *Identity Server: External Attribute Source* for the type of policy, then click *OK*.

4. Fill in the following fields:

   Description: (Optional) Describe the purpose of this policy.

   Priority: Specify the order in which a rule is applied in the policy, when the policy has multiple rules. The highest priority is 1 and the lowest priority is 10.

5 In the *Actions* section, click *New*, then select *Fetch Attributes*.

6 Specify the following details:

External Attribute Name: Specify the name of the attribute to be obtained through this policy.

Value: Specify either *String Constant* or *Data Extension* for the attribute value.

If you select *String Constant*, provide the value in the text box. The policy returns the string constant.

If you select *Data Extension*, select the extension file from the list. The policy returns the attributes based on the logic defined in the class.

For more information about policy extension, see Section 1.7, "Adding Policy Extensions," on page 16.

7 To save the policy, click *OK* twice, then click *Apply Changes*.

8 After creating an External Attribute Source policy, create a shared secret. This shared secret is used in configuring other policies or can be used by the Identity Servers in their attribute sets to retrieve attributes from external sources.

For more information, see "Creating Shared Secret Names" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

7 Troubleshooting Access Manager Policies

This section discusses the following topics:

- Section 7.1, "Turning on Logging for Policy Evaluation," on page 161
- Section 7.2, "Understanding Policy Evaluation Traces," on page 162
- Section 7.3, "Common Configuration Problems That Prevent a Policy from Being Applied as Expected," on page 180
- Section 7.4, "The Policy Is Using Old User Data," on page 182
- Section 7.5, "Form Fill and Identity Injection Silently Fail," on page 183
- Section 7.6, "Checking for Corrupted Policies," on page 184
- Section 7.7, "Policy Page Timeout," on page 184
- Section 7.8, "Policy Creation and Storage," on page 184
- Section 7.9, "Policy Distribution," on page 184
- Section 7.10, "Policy Evaluation: Access Gateway Devices," on page 185

7.1 Turning on Logging for Policy Evaluation

Policy evaluation for roles occurs at the Identity Server. For Authorization and Identity Injection policies, policy evaluation occurs on the Embedded Service Provider (ESP) where the policy is enabled.

For the Form Fill policies, the evaluation and logging is done by ESP and the proxy service. To set the logging level on the Access Gateway for the proxy service, see "Enabling Form Fill Logging" on page 175.

Logging for the policy evaluation done by ESP is controlled by the log settings of the Identity Server configuration. To enable this type of logging:

1. Click *Devices* > *Identity Servers* > *Edit* > *Logging*.

   If you have set up more than one Identity Server configuration, make sure you select the configuration to which the other Access Manager components have been assigned.
2. Select *Enabled* for *File Logging*.
3. Select to echo the trace messages to the console:
   - For the Linux Access Gateway Appliance, Linux Access Gateway Service, or Linux Identity Server, this sends the messages to the `catalina.out` file.
   - For the Linux Access Gateway Service or Windows Identity Server, this sends the messages to the `stdout.log` file.
4. (Optional) Specify a path for the Identity Server log files.

If you have a mixed platform environment (for example, the Identity Server is installed on Windows and the Access Gateway is on Linux), do not specify a path.

5 For policy evaluation tracing, set the *Application* level to *info* in the *Component File Logger Levels* section.

If you are only troubleshooting policies at this time, do not select any other options. This reduces the amount of information recorded in the log files.

To see the policy SOAP messages, you need to set the *Application* level to *config.*

6 Update the Identity Server.

7 Click *Auditing* > *General Logging*.

- For role evaluation traces, view the Identity Server `catalina.out` file (Linux) or the `stdout.log` file (Windows).

  If your Identity Servers are clustered, you need to look at the file from each Identity Server.

- For Authorization, Form Fill, and Identity Injection evaluation traces, view the log file of ESP of the device that is protecting the resource.

  - **Linux Access Gateway Appliance or Service:** This is the `catalina.out` file of the Access Gateway where the protected resource is defined. If the Access Gateway is part of a cluster, you need to look at this file from each Access Gateway in the group.

    To view the actual ESP log file that contains only ESP log messages, see the `nidp.*.xml` files in the `/var/opt/novell/tomcat7/webapps/nesp/WEB-INF/logs` directory (or the directory you specified in Step 4). Depending upon how you have configured *File Wrap*, the * portion of the filename contains the month, the week, the day, and the hour.

  - **Windows Access Gateway Service:** This is the `stdout.log` file of the Access Gateway where the protected resource is defined. If the Access Gateway is part of a cluster, you need to look at this file from each Access Gateway in the group.

    To view the actual ESP log file that contains only ESP log messages, see the `nidp.*.xml` files in the `\Program Files\Novell\tomcat\webapps\nesp\WEB-INF\logs` directory (or the directory you specified in Step 4). Depending upon how you have configured *File Wrap*, the * portion of the filename contains the month, the week, the day, and the hour.

  - **J2EE Agent:** See "Viewing Log Files" in the *NetIQ Access Manager 3.2 J2EE Agent Guide*.

8 To understand what you are looking for in the log file, continue with one of the following:

- Section 7.2, "Understanding Policy Evaluation Traces," on page 162 if you set *Application* level to *info.*
- Section 7.10, "Policy Evaluation: Access Gateway Devices," on page 185 if you set *Application* level to *config*.

7.2 Understanding Policy Evaluation Traces

- Section 7.2.1, "Format," on page 163
- Section 7.2.2, "Policy Result Values," on page 169
- Section 7.2.3, "Role Assignment Traces," on page 170
- Section 7.2.4, "Identity Injection Traces," on page 171
- Section 7.2.5, "Authorization Traces," on page 173
- Section 7.2.6, "Form Fill Traces," on page 175

7.2.1 Format

A policy log entry starts with the standard log entry elements: `<amLogEntry>` followed by the correlation tags. (For information about correlation tags, see "Understanding the Correlation Tags in the Log Files" in the *NetIQ Access Manager 3.2 Administration Console Guide*.) The following log entry is a trace of an evaluation of a Role policy:

```
<amLogEntry> 2009-06-07T21:40:25Z INFO NIDS Application: AM#500199050:
AMDEVICEID#9921459858EAAC29: AMAUTHID#503EFFA4BC21ACA307796EC7D96E5532: IDP
RolesPep.evaluate(), policy trace:
   ~~RL~0~~~~Rule Count: 1~~Success(67)
   ~~RU~RuleID_1181251958207~Manager~DNF~~1:1~~Success(67)
   ~~CS~1~~ANDs~~1~~True(69)
   ~~CO~1~LdapGroup(6645):no-param:hidden-value:~ldap-group-is-member-
of~SelectedLdapGroup(66455):hidden-param:hidden-value:~~~True(69)
   ~~PA~ActionID_1181252224665~~AddRole~Manager~~~Success(0)
   ~~PC~ActionID_1181252224665~~Document=(ou=xpemlPEP,ou=mastercdn,
ou=ContentPublisherContainer,ou=Partition,ou=PartitionsContainer,ou=VCDN_Root,ou=a
ccessManagerContainer,o=novell:romaContentCollectionXMLDoc),Policy=(Manager),Rule=
(1::RuleID_1181251958207),Action=(AddRole::ActionID_1181252224665)~AdditionalRole(
6601):unknown():Manager:~~~Success(0)
 </amLogEntry>
```

The Role policy evaluated in this entry has the following definition:

Figure 7-1 *Manager Policy Definition*

The following sections use this policy and its trace to explain the information contained within each line of a policy trace. The policy trace part of the entry starts with a `policy trace:`, which is followed by one or more of the following types:

- RL - Rule List Evaluation Result
- RU - Rule Evaluation Result

- CS - Condition Set Evaluation Result
- CO - Condition Evaluation Result
- PA - Policy Action Initiation
- PC - Policy Action Completion

Elements within a type are separated from each other with the tilde (~) character. If an element does not have a value, no value is inserted, which results in two or more tildes between values. Two tildes means one element didn't have a value, three tildes means that two elements didn't have values, and so forth.

Rule List Evaluation Result

An RL trace has the following fields:

```
~<RuleListID>~~~~<RuleCount>~~~<Result>
```

A RL trace looks similar to the following:

```
~~RL~1~~~~Rule Count: 1~~Success(67)
```

Table 7-1 describes the fields found in an RL trace.

Table 7-1 *Fields in a Rule List Trace*

Element	Description
`<RuleListID>`	The identifier assigned to the rule list. In the sample RL trace, this is 1.
`<RuleCount>`	The number of rules defined for the policy. In the sample RL trace, this is `Rule Count: 1`, indicating that there is one rule in the policy.
`<Result>`	A string followed by a number that specifies the result of the evaluation. See "Policy Result Values" on page 169. In the sample RL trace, this is `Success(67)`, indicating success.

Rule Evaluation Result

An RU trace has the following fields:

```
~<RuleID>~<ParentPolicyName>~<ConditionSetJoinType>~~~<ConditionSetCount:
ActionCount>~~~<Result>
```

An RU trace looks similar to the following:

```
~~RU~RuleID_1181251958207~Manager~DNF~~1:1~~Success(67)
```

Table 7-2 describes the fields of a Rule Evaluation Result trace.

Table 7-2 Fields in a Rule Evaluation Result Trace

Element	Description
`<RuleID>`	The identifier assigned to the rule. In this sample RU trace, this element is set to `RuleID_1181251958207`.
`<ParentPolicyName>`	The name of the parent policy to which the rule is assigned. In this sample RU trace, this element is set to `Manager`.
`<ConditionSetJoinType>`	The type of joining that occurs between conditions and condition sets. It is set to one of the following: ◆ **CNF:** Indicates that sets are ANDed and conditions within a condition group are ORed. ◆ **DNF:** Indicates that sets are ORed and conditions within a condition group are ANDed. In the sample RU trace, this element is set to `DNF`.
`<ConditionSetCount:ActionCount>`	The number of condition sets and actions defined for this rule. In the sample RU trace, this is 1:1, for one condition set and one action.
`<Result>`	A string followed by a number that specifies the result of the evaluation. See "Policy Result Values" on page 169. In the sample RU trace, this is `Success(67)`, indicating that the rule was successfully evaluated.

Condition Set Evaluation Result

A CS trace has the following fields

```
~<ConditionSetID>~<JoinType>~<NOT>~<ConditionCount>~~~<Result>
```

A CS trace looks similar to the following:

```
~~CS~1~~ANDs~~1~~True(69)
```

Table 7-3 describes the fields in a Condition Set trace.

Table 7-3 Fields in a Condition Set Trace

Element	Description
`<ConditionSetID>`	The identifier assigned to the condition set. Rules can have multiple condition sets. In this sample CS trace, this is 1, for the first and only condition set defined for the rule.
`<JoinType>`	Specifies how the condition results are combined, if there are multiple condition sets. Possible values include `ANDs` and `ORs`. In this sample CS trace, this is `ANDs`.

Element	Description
`<NOT>`	The string `NOT` if the result was negated prior to reporting; otherwise the field has no value. This is the *If Not* option when creating a condition group. In the sample CS trace, the condition group was not negated, therefore the field is not present.
`<ConditionCount>`	The number of conditions defined in the condition group. In the sample CS trace, this element has the value of 1.
`<Result>`	A string followed by a number that specifies the result of the evaluation. See "Policy Result Values" on page 169. In the sample CS trace, this is True (69), indicating that the condition evaluated to True.

Condition Evaluation Result

A CO trace has the following fields:

```
~<ConditionID>~<LHSOperand>~<Operator>~<RHSOperand>~<NOT>~<Result>[~<ResultOnError
>]
```

A CO trace looks similar to the following:

```
~~CO~1~LdapGroup(6645):no-param:hidden-value:~ldap-group-is-member-
of~SelectedLdapGroup(66455):hidden-param:hidden-value:~~~True(69)
```

Table 7-4 describes the fields in a Condition trace.

Table 7-4 *Fields in a Condition Trace*

Element	Description
`<ConditionID>`	The identifier assigned to the conditions in the condition group. The first condition is assigned 1. In the sample CO trace, this is 1.

Element	Description
`<LHSOperand>`	The enumerative value and parameter list of the left operand. It is the first value specified for the comparison and has the following format: `<Condition Name(Data ID)>: <Parameter> : <Value>` The Condition Name is the string assigned to the condition type specified in the policy. The Data ID is a numerical value assigned to the condition type. `<Parameter>` contains one of the following strings: ◆ `no-param` when no parameters are specified for the operand, followed by a colon, followed by one of the following: the value, `no-value`, or `hidden-value` when the value contains sensitive information. ◆ `hidden-param` followed by a colon, and then `hidden-value`. This string is used when both the parameter and its value contain sensitive information. In the sample CO trace, this is `LdapGroup(6645):no-param:hidden-value`. LdapGroup is the string for the LDAP Group condition. The policy specified *[Current]*, so no parameters were specified. The groups that the user belongs to are considered sensitive data, so the log file displays `hidden-value` for the names of the groups.
`<Operator>`	The display name of the comparison operator. In the sample CO trace, this is `ldap-group-is-member-of`. In the policy, this is displayed as *LDAP Group: Is Member of*.
`<RHSOperand>`	The enumerative value and parameter list of the right operand. It is the second value specified for the comparison and has the same format as the `<LHSOperand>`. In the sample CO trace, this is `SelectedLdapGroup(66455):hidden-param:hidden-value`. The actual policy specifies LDAP Group as the parameter, and the value is the DN of the group.
`<NOT>`	The string `NOT` if the result was negated prior to reporting; otherwise the field has no value. This is the *If Not* option when creating a condition. In the sample CO trace, this condition result was not negated, therefore the field is represented by a tilde.
`<Result>`	A string followed by a number that specifies the result of the comparison. See "Policy Result Values" on page 169. In the sample CO trace, this is True (69), indicating that the condition evaluated to True—the user is a member of the specified LDAP group.
`<ResultOnError>`	A string describing the error that occurred. This is an optional field that only appears when the condition evaluation results in an error. The sample CO trace did not result in an error, so it has no string.

Policy Action Initiation

A PA trace has the following fields:

```
~<ActionID>~<TraceString1>~<TraceString2>~<TraceString3>~<Result>
```

A PA trace looks similar to the following:

```
~~PA~ActionID_1181252224665~~AddRole~Manager~~~Success(0)
```

Table 7-5 describes the fields in a Policy Action trace.

Table 7-5 *Fields in a Policy Action Trace*

Element	Description
`<ActionID>`	The identifier assigned to the action. In the sample PA trace, this is `ActionID_1181252224665`.
`<TraceString1>`	The message specified with the action. In the sample PA trace, this is `AddRole`.
`<TraceString2>`	The second part of the specified message. In the sample PA trace, this is `Manager`.
`<TraceString3>`	The third part of the specified message. In the sample PA trace, this field has no value and is not present.
`<Result>`	A string followed by a number that specifies the result of the assigning the action. See "Policy Result Values" on page 169. In the sample PA trace, this is `Success(0)`, which indicates that the action of assigning the Manager role to the user was successful.

Policy Action Completion

A PC trace has the following fields

```
~<ActionID>~<ActionName>~<ActionParmeters>~~~<Result>[~<ActionError>]
```

A PC trace looks similar to the following:

```
~~PC~ActionID_1181252224665~~Document=(ou=xpemlPEP,ou=mastercdn,
ou=ContentPublisherContainer,ou=Partition,ou=PartitionsContainer,ou=VCDN_Root,ou=a
ccessManagerContainer,o=novell:romaContentCollectionXMLDoc),Policy=(Manager),Rule=
(1::RuleID_1181251958207),Action=(AddRole::ActionID_1181252224665)~AdditionalRole(
6601):unknown():Manager:~~~Success(0)
```

Table 7-6 describes the fields in a Policy Action Completion trace.

Table 7-6 *Fields in a Policy Action Completion Trace*

Element	Description
`<ActionID>`	The ID assigned to the action. In the sample PC trace, this is `ActionID_1181252224665`.

Element	Description
`<ActionName>`	The fully distinguished name of the action. In the sample PC trace, the action has the following parts in its name: ◆ Document=(ou=xpemlPEP,ou=mastercdn, ou=ContentPublisherContainer,ou=Partition,ou=PartitionsContainer,ou=VCDN_Root,ou=accessManagerContainer,o=novell:romaContentCollectionXMLDoc) ◆ Policy=(Manager) ◆ Rule=(1::RuleID_1181251958207) ◆ Action=(AddRole::ActionID_1181252224665)
`<ActionParmeters>`	A list of the action parameters passed to the action handler. In this sample PC trace, the Role policy has an action and a parameter. The value of this element is `AdditionalRole(6601):unknown(): Manager:`
`<Result>`	A string followed by a number that specifies the result. See "Policy Result Values" on page 169. In the sample PC trace, this is `Success(0)`and indicates success.
`<ActionError>`	A string describing the error that occurred when invoking the action. This is an optional field that only appears when the Result field contains an error code. The sample PC trace did not result in an error, so it has no string.

7.2.2 Policy Result Values

The last field in a trace string is the `<result>` field. Table 7-7 lists the possible values:

Table 7-7 *Result Values from Policy Traces*

Value	Name	Description
0	Success	The policy evaluation was successful.
1	Error: No memory	The system is out of memory.
2	Error: Bad data	The data sent for evaluation is invalid.
3	Error: Configuration initialization	An error was detected during the policy configuration processing.
4	Error: General failure	An error was detected during policy processing.
5	Pending	The policy processing is in progress.
64	Permit	The rule produced a Permit action.
65	Deny	The rule produced a Deny action.
66	Obligation	The rule triggered an obligation, indicating that additional processing is required. Identity Injection policies trigger obligations.
67	No action	The rule did not initiate any action.

Value	Name	Description
68	Condition false	The condition evaluated to False.
69	Condition true	The condition evaluated to True.
70	Condition unknown	Condition input was not available, so the results are unknown.
71	Cancel	The current operation has been canceled.
72	Error: Interface unavailable	The current operation is unavailable.
73	Error: Data unavailable	The data required for evaluation was unavailable.
74	Error: Illegal state	Processing error; report it to Novell® Support.

7.2.3 Role Assignment Traces

The following sections walk you through a few sample role traces. When you understand these traces, you should be able to understand any role trace.

- "When the User Is Assigned Roles" on page 170
- "When the Role Policy Is Not Enabled" on page 171
- "When an Authorization Policy Uses a Role" on page 171

When the User Is Assigned Roles

Roles are assigned at authentication, so this type of trace is found in the `catalina.out` file (Linux) or the `stdout.log` file (Windows) of the Identity Server. This is a trace of a user who does not match the requirements to be assigned the Manager Role (for a definition of this Role policy, see Figure 7-1 on page 163).

```
<amLogEntry> 2009-06-11T15:38:38Z INFO NIDS Application: AM#500199050:
AMDEVICEID#9921459858EAAC29: AMAUTHID#0CE611AAE4D0301F26DD4865476BDA1 4: IDP
RolesPep.evaluate(), policy trace:
   ~~RL~0~~~~Rule Count: 1~~Success(67)
   ~~RU~RuleID_1181251958207~Manager~DNF~~1:1~~Success(67)
   ~~CS~1~~ANDs~~1~~False(68)
   ~~CO~1~LdapGroup(6645):no-param:hidden-value:~ldap-group-is-member-
of~SelectedLdapGroup(66455):hidden-param:hidden-value:~~~False(68)
</amLogEntry>
```

This trace describes the following about the policy.

1. The RL trace indicates that the policy has one rule and that the policy evaluated without error.
2. The RU trace indicates that the rule (`RuleID_1181251958207`) has one condition and one action and that the rule evaluated without error.
3. The CS trace indicates that the condition set evaluated to False (the user logging in does not match the conditions of the set).
4. The CO trace indicates that the condition evaluated to False (the user logging in does not match the condition).

When you are troubleshooting why a user is not granted access to a resource that uses a role in its Authentication policy, the first step should be to look at the Identity Server file and determine whether the user was assigned the role. In this trace, you can see that the user was not assigned the role. To fix this problem, you can either change the conditions of the Role policy to match the user or change the user's information so that the user matches the existing condition in the Role policy.

When the Role Policy Is Not Enabled

Sometimes a Role policy is created, but the Role policy is not enabled for the Identity Server. When this happens, the trace looks similar to the following:

```
<amLogEntry> 2009-06-11T16:06:03Z INFO NIDS Application: AM#500199050:
AMDEVICEID#9921459858EAAC29: AMAUTHID#FDE680ABE320B682038947EA5F59D6B F: IDP
RolesPep.evaluate(), policy trace:
   ~~RL~0~~~~Rule Count: 0~~Success(67)
 </amLogEntry>
```

When you see Role policy traces that contain only the RL trace line, you need to enable the Role policy.

When an Authorization Policy Uses a Role

When a user requests access to a resource that has an Authorization policy that uses a role, the user is checked for the role assignment. The trace of this evaluation is in the ESP log file of the Access Gateway that is processing the request. Such a trace looks similar to the following:

```
<amLogEntry> 2009-07-13T22:13:29Z INFO NIDS Application: AM#501102050:
AMDEVICEID#esp-51A474B83BFDDF4F: AMAUTHID#4538DB6F6E2A237FDE674F0C6E1 6DCEC:
PolicyID#N748097P-3507-3KP7-4241-410PN4152094: NXPESID#1718: AGAuthorization
Policy Trace:
   ~~RL~1~~~~Rule Count: 1~~Success(0)
   ~~RU~RuleID_1182876316974~Allow_Sales~DNF~~1:1~~Success(0)
   ~~CS~1~~ANDs~NOT~1~~True(69)
   ~~CO~1~CurrentRoles(6660):no-param:authenticated~com.novell.nxpe.
condition.NxpeOperator@string-substring~SelectedRole(6661):hidden-param:hidden-
value:~~~False(68)
   ~~PA~1~~Deny Access Messasge~Sorry, you must work in sales today.~~~Success(0)
   ~~PC~1~~Document=(ou=xpemlPEP,ou=mastercdn,ou=ContentPublisherCon
tainer,ou=Partition,ou=PartitionsContainer,ou=VCDN_Root,ou=accessManagerContainer,
o=novell:romaContentCollectionXMLDoc),Policy=(Allow_Sales),Rule=(1::RuleID_1182876
316974),Action=(Deny::1)~~~~Success(0)
</amLogEntry>
```

This trace is for a Deny policy that denies access if the user has not been assigned the Sales role. The CO line indicates that the condition is looking for a role and that the user did not match the condition.

The CS line indicates that the condition is a negative condition, meaning that the user matches the condition set when the user does not match the condition. This is the case for this user, so the condition set evaluates to True, and the action is then applied.

The PA line describes the action that was applied.

7.2.4 Identity Injection Traces

The following traces explain what to look for in an Identity Injection policy that injects an authorization header:

- "When the User Has Authenticated" on page 172
- "When the User Hasn't Authenticated" on page 172

When the User Has Authenticated

The following trace is for an Identity Injection policy that successfully inserts an authentication header. The policy inserts LDAP credentials for the user's name and password. The Access Gateway injects the information, so the trace for this type of policy is in the ESP log file of the Access Gateway.

```
<amLogEntry> 2009-06-11T19:02:44Z INFO NIDS Application: AM#501103050:
AMDEVICEID#esp-534FD0D0E32FE4BD: AMAUTHID#61D5D5B3FF98156F8E4F2875981D 4A6E:
PolicyID#51N4214K-74L1-491L-7190-2M9K04K21393: NXPESID#726: AGIdentityInjection
Policy Trace:
   ~~RL~0~~~~Rule Count: 1~~Success(67)
   ~~RU~RuleID_1181251426062~basic_auth_ii~DNF~~0:1~~Success(67)
   ~~PA~ActionID_1181251427701~~Inject Auth Header~uid~uid(1):
CredentialProfile(7010:):NEPXurn~3Anovell~3Acredentialprofile~3A2005-
03~2Fcp~3ASecrets~2Fcp~3ASecret~2Fcp~3AEntry~40~40~40~40WSCQSSToken~40~40~40~40~2F
cp~3ASecrets~2Fcp~3ASecret~5Bcp~3AName~3D~22LDAPCredentials~22~5D~2Fcp~3AEntry~5Bc
p~3AName~3D~22UserName~22~5D:~Ok~Success(0)
   ~~PA~ActionID_1181251427701~~Inject Auth Header~password~pwd(1):
CredentialProfile(7010:):NEPXurn~3Anovell~3Acredentialprofile~3A2005-
03~2Fcp~3ASecrets~2Fcp~3ASecret~2Fcp~3AEntry~40~40~40~40WSCQSSToken~40~40~40~40~2F
cp~3ASecrets~2Fcp~3ASecret~5Bcp~3AName~3D~22LDAPCredentials~22~5D~2Fcp~3AEntry~5Bc
p~3AName~3D~22UserPassword~22~5D:~Ok~Success
(0)
   ~~PC~ActionID_1181251427701~~Document=(ou=xpemlPEP,ou=mastercdn,
ou=ContentPublisherContainer,ou=Partition,ou=PartitionsContainer,ou=VCDN_Root,ou=a
ccessManagerContainer,o=novell:romaContentCollectionXMLDoc),Policy=(basic_auth_ii)
,Rule=(1::RuleID_1181251426062),Action=(InjectAuthHeader::ActionID_1181251427701)~
~~~Success(0)
 </amLogEntry>

<amLogEntry> 2009-06-11T19:02:44Z INFO NIDS Application: AM#501101021:
AMDEVICEID#esp-534FD0D0E32FE4BD: AMAUTHID#61D5D5B3FF98156F8E4F2875981D 4A6E:
PolicyID#51N4214K-74L1-491L-7190-2M9K04K21393: NXPESID#726: Response sent: Status
- success </amLogEntry>
```

Each identity injection policy generates two log entries. The first entry indicates whether the policy could successfully retrieve the information and inject it into the header. The second entry specifies whether the response is successfully sent to the Web server.

This first log entry describes the following about this policy:

1. In the correlation tags (AM... tags), notice the ID assigned to the authenticated user making the request (`AMAUTHID#61D5D5B3FF98156F8E4F2875981D4A6E`).
2. After the correlation tags, the trace specifies the ID of the policy (`51N4214K-74L1-491L-7190-2M9K04K21393`).
3. The RU trace indicates that the policy name is basic_auth_ii, that the policy has no conditions, and that the policy has one action rule.
4. The first PA trace indicates that the uid (called LDAP User Name in the UI) of the Credential Profile has been successfully retrieved.
5. The second PA trace indicates that the password of the Credential Profile has been successfully retrieved.
6. The PC trace indicates that these items have been successfully injected into the header.

You can use the user's ID and the policy ID to find log entry that traces the response to the Web server. The second log entry indicates that the response was successfully sent to the Web server.

When the User Hasn't Authenticated

If the user has not authenticated and therefore has no authentication credentials, the trace for an Identity Injection policy with an authentication header looks similar to the following:

```
<amLogEntry> 2009-06-11T20:16:51Z INFO NIDS Application: AM#501103050:
AMDEVICEID#esp-534FD0D0E32FE4BD: PolicyID#OL8659PL-0K69-0N0N-0845-5PN113KM3842:
NXPESID#2539: AGIdentityInjection Policy Trace:
   ~~RL~0~~~~Rule Count: 1~~Success(67)
   ~~RU~RuleID_1181251426062~basic_auth_ii~DNF~~0:1~~Success(67)
   ~~PA~ActionID_1181251427701~~Inject Auth Header~uid~uid(1):
CredentialProfile(7010:):NEPXurn~3Anovell~3Acredentialprofile~3A2005-
03~2Fcp~3ASecrets~2Fcp~3ASecret~2Fcp~3AEntry~40~40~40~40WSCQSSToken~40~40~40~40~2F
cp~3ASecrets~2Fcp~3ASecret~5Bcp~3AName~3D~22LDAPCredentials~22~5D~2Fcp~3AEntry~5Bc
p~3AName~3D~22UserName~22~5D:~Ok~Success(0)
   ~~PA~ActionID_1181251427701~~Inject Auth
Header~password~pwd(1):CredentialProfile(7010:):NEPXurn~3Anovell~3Acredentialprofi
le~3A2005-03~2Fcp~3ASecrets~2Fcp~3ASecret~2Fcp~3AEntry
~40~40~40~40WSCQSSToken~40~40~40~40~2Fcp~3ASecrets~2Fcp~3ASecret~5Bcp~3AName~3D~22
LDAPCredentials~22~5D~2Fcp~3AEntry~5Bcp~3AName~3D~22UserPassword~22~5D:~Ok~Success
(0)
   ~~PC~ActionID_1181251427701~~Document=(ou=xpemlPEP,ou=mastercdn,
ou=ContentPublisherContainer,ou=Partition,ou=PartitionsContainer,ou=VCDN_Root,ou=a
ccessManagerContainer,o=novell:romaContentCollectionXMLDoc),Policy=(basic_auth_ii)
,Rule=(1::RuleID_1181251426062),Action=(InjectAuthHeader::ActionID_1181251427701)~
~~~Success(0)
 </amLogEntry>

<amLogEntry> 2009-06-11T20:16:51Z INFO NIDS Application: AM#501101021:
AMDEVICEID#esp-534FD0D0E32FE4BD: PolicyID#OL8659PL-0K69-0N0N-0845-5PN113KM3842:
NXPESID#2539: Response sent: Status - success </amLogEntry>
```

These entries look very similar to the entries for a successful injection of data. This is because injecting NULL data for data that is not available is considered a successful action. The trace displays data unavailable errors only when errors occur retrieving data. The key to determining whether the data was available for injection into an authentication header is to look for the `AMAUTHID` correlation tag in the log entry. The log entries for the `OL8659PL-0K69-0N0N-0845-5PN113KM3842` policy do not contain an `AMAUTHID` correlation tag, which indicates that the user is not logged in.

7.2.5 Authorization Traces

Authorization policies for a protected resource might require a user to be authenticated before the data required by the policy can be obtained, but Authorization policies can be configured to use data that is available without authentication. The following traces show how the log entries for an Authorization policy trace are slightly different when the user is not authenticated.

- "When the Protected Resource Requires Authentication" on page 173
- "When the Protected Resource Does Not Require Authentication" on page 174

For a trace of an Authorization policy that uses a role, see "When an Authorization Policy Uses a Role" on page 171.

When the Protected Resource Requires Authentication

The following is a successful trace of an Authorization policy that requires the user to have the value of Manager in the title attribute. To obtain this data, the user must be authenticated.

The policy contains two rules: a Permit rule if the user has the value of Manager in the title attribute, and a Deny rule that denies all other users. This policy has been assigned to protect an Access Gateway resource.

```
<amLogEntry> 2009-08-02T15:55:05Z INFO NIDS Application: AM#501101050:
AMDEVICEID#esp-2FA73CE1A376FD91: PolicyID#459O8443-N8P5-KO21-68OM-K172P107N4O5:
NXPESID#1743: Evaluating policy </amLogEntry>

<amLogEntry> 2009-08-02T15:55:06Z INFO NIDS Application: AM#501102050:
AMDEVICEID#esp-2FA73CE1A376FD91: AMAUTHID#838976482579AF372C31C4727 4E9CB28:
PolicyID#459O8443-N8P5-KO21-68OM-K172P107N4O5: NXPESID#1743: AGAuthorization
Policy Trace:
   ~~RL~1~~~~Rule Count: 2~~Success(0)
   ~~RU~RuleID_1186068489688~Title_auth~DNF~~1:1~~Success(0)
   ~~CS~1~~ANDs~~1~~True(69)
   ~~CO~1~LdapAttribute(6647):NEPXurn~3Anovell~3Aldap~3A2006-
02~2Fldap~3AUserAttribute~40~40~40~40WSCQLDAPToken~40~40~40~40~2FUserAttribute~5B~
40ldap~3AtargetAttribute~3D~22title~22~5D:hidden-
value:~com.novell.nxpe.condition.NxpeOperator@string-equals~(0):hidden-
param:hidden-value:~~~True(69)
   ~~PA~1~~Permit Access~~~~Success(0)
   ~~PC~1~~Document=(ou=xpemlPEP,ou=mastercdn,ou=ContentPublisher
Container,ou=Partition,ou=PartitionsContainer,ou=VCDN_Root,ou=accessManagerContain
er,o=novell:romaContentCollectionXMLDoc),Policy=(Title_auth),Rule=(1::RuleID_11860
68489688),Action=(Permit::1)~~~~Success(0)
</amLogEntry>

<amLogEntry> 2009-08-02T15:55:06Z INFO NIDS Application: AM#501101021:
AMDEVICEID#esp-2FA73CE1A376FD91: AMAUTHID#838976482579AF372C31C47274E 9CB28:
PolicyID#459O8443-N8P5-KO21-68OM-K172P107N4O5: NXPESID#1743: Response sent: Status
- success </amLogEntry>
```

The first log entry is the request to evaluate the policy. The second log entry is the evaluation of the policy. The third log entry is the response that is returned. These three log entries can be tied together by using the following tags:

AMDEVICEID#esp-2FA73CE1A376FD91: When a policy evaluation request is made, the same ESP processes the request. Even if the Access Gateways are clustered, the policy evaluation request stays with the Access Gateway that initiated the request.

PolicyID#459O8443-N8P5-KO21-68OM-K172P107N4O5: Each policy is assigned a unique ID, and this is the ID assigned to the policy called Title_auth in the Administration Console. To search for all log entries for a policy, use the policy ID. To search for log entries that evaluate the policy, use the policy name.

AMAUTHID#838976482579AF372C31C47274E9CB28: The request to evaluate a policy does not contain the ID of the user the request is being made for, but the log entries for the evaluation and for the response status always contain the ID of an authenticated user. If the policy can be evaluated without the user being authenticated, these entries do not contain the ID of the user. This kind of policy might be assigned to a public resource (no authentication required) and use the time of day condition or day of the week condition for its evaluation criteria. See "When the Protected Resource Does Not Require Authentication" on page 174.

When the Protected Resource Does Not Require Authentication

The following trace is for an Authorization policy that uses data that is available without authentication. Authorization policies support a number of these conditions, such as Current Date, Current Day of Week, Current Day of Month, Current Time Of Day, Client IP, and the URL conditions. As long as you do not select to compare what is currently in the HTTP request with a value that requires authentication (such as LDAP attribute), the Authorization policy can be evaluated for an unauthenticated user. The following trace is for a policy with a Current Time of Day condition. The protected resource does not require authentication, so everyone can access the resource if their request comes in between 8:00 am and 5:30 pm, local time.

```
<amLogEntry> 2009-08-03T16:30:48Z INFO NIDS Application: AM#501101050:
AMDEVICEID#esp-2FA73CE1A376FD91: PolicyID#216660PM-429P-O660-N25N-L58L08MN4N5M:
NXPESID#4515: Evaluating policy </amLogEntry>

<amLogEntry> 2009-08-03T16:30:48Z INFO NIDS Application: AM#501102050:
AMDEVICEID#esp-2FA73CE1A376FD91: PolicyID#216660PM-429P-O660-N25N-L58L08MN4N5M:
NXPESID#4515: AGAuthorization Policy Trace:
   ~~RL~1~~~~Rule Count: 2~~Success(0)
   ~~RU~RuleID_1186082720202~time_of_day~DNF~~1:1~~Success(0)
   ~~CS~1~~ANDs~~1~~True(69)
   ~~CO~0~TimeOfDay(1005):::Fri Aug 03 10:30:48 MDT
2007(9:30):~com.novell.nxpe.condition.NxpeOperator@time-in-
range~(0)::::~~~True(69)
   ~~PA~1~~Permit Access~~~~Success(0)
   ~~PC~1~~Document=(ou=xpemlPEP,ou=mastercdn,ou=ContentPublisherCon
tainer,ou=Partition,ou=PartitionsContainer,ou=VCDN_Root,ou=accessManagerContainer,
o=novell:romaContentCollectionXMLDoc),Policy=(time_of_day),Rule=(1::RuleID_1186082
720202),Action=(Permit::1)~~~~Success(0)
 </amLogEntry>

<amLogEntry> 2009-08-03T16:30:48Z INFO NIDS Application: AM#501101021:
AMDEVICEID#esp-2FA73CE1A376FD91: PolicyID#216660PM-429P-O660-N25N-L58L08MN4N5M:
NXPESID#4515: Response sent: Status - success </amLogEntry>
```

The first log entry is the request to evaluate the policy. The second log entry is the evaluation of the policy, and from it you can tell that the user is not authenticated because the `AMAUTHID#` tag is missing. The third log entry is the response that is returned, and it indicates that a success was returned. The user is allowed access to the resource.

7.2.6 Form Fill Traces

The following sections describe how to enable logging for the Form Fill policies, describe the form that was used to create the Form Fill trace, then describe the entries that can be found in the logs:

- "Enabling Form Fill Logging" on page 175
- "Sample Form and Policy Used for the Trace" on page 175
- "Embedded Service Provider Trace" on page 178
- "Proxy Service Trace" on page 178

Enabling Form Fill Logging

Two modules evaluate the Form Fill policy and log entries:

- Embedded Service Provider (ESP) of the Access Gateway evaluates the Form Fill policy and logs entries to its file. ESP sends the messages to the `catalina.out` file (Linux) or the `stdout.log` file (Windows) of the Access Gateway. To enable ESP logging, see Section 7.1, "Turning on Logging for Policy Evaluation," on page 161.
- The proxy service of the Access Gateway reports on the process of finding the form data and filling it in.

  For the Access Gateway Appliance, see the `/var/log/lagsoapmessages` file.

  You can configure a custom filter and file to log Form Fill entries. For the filter, enable the *Form Fill Processing* events in the *Advanced Log Level Options* section.

Sample Form and Policy Used for the Trace

Figure 7-2 illustrates the simple form that was used for the trace.

Figure 7-2 *Form Used for the Trace*

Source HTML for the Form

The name of the form and the fields that need to be filled in by the policy are in bold typeface.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta http-equiv="Content-type" content="text/html; charset=utf-8">
  <title>kelly</title>
</head>
<body>
  <form name="mylogin" action="double.php" method="post" id="mylogin">
    <center>
      <table border="0" cellpadding="4" cellspacing="4" width="570">
        <tr>
          <td width="121" height="285" align="left" valign="top">
          </td>
          <td width="449" height="285" align="center" valign="top">
            <p align="center">
              <font size="5">Novell Services Login<br></font>
            </p>
            <table border="0" width="86%">
              <tr>
                <td width="25%">Username:</td>
                <td width="75%">
                  <input type="TEXT" name="username">
                </td>
              </tr>
              <tr>
                <td width="25%">Password:</td>
                <td width="75%">
                  <input type="PASSWORD" name="password" size="30">
                </td>
              </tr>
              <tr>
                <td width="25%">title:</td>
                <td width="75%">
                  <input type="TEXT" name="title" size="30">
                </td>
              </tr>
            </table>
          </td>
```

```
        </tr>

         <tr>
            <td colspan="2" align="center">
             <input type="hidden" name="formNum" value="1">
             <input type="submit" value="Login">
             <input type="reset">
            </td>
          </tr>
        </table>
      </center>
    </form>
</body>
</html>
```

Form Fill Policy

The following Form Fill policy was created for the `mylogin` form. The policy is called `simpleform`. You can use the name of the policy to find entries for it in the log files. The policy was assigned to the `/identity/forms/simple.html` protected resource. Because the URL path identifies a specific file on the Web server, the policy does not require any CGI or page matching criteria.

Figure 7-3 *The Form Fill Policy for the mylogin Form*

This policy is configured so that the user never sees the form. Even on first login, the form is filled in for authenticated users because the user's authentication credentials are used for the username and password fields, and the title field value is obtained from the LDAP user store. If the user does not have a value for the title attribute, the user sees the form every time the page is accessed. If you want the value to be saved for these users, you need to change the policy to use a secret store rather than an LDAP attribute.

Embedded Service Provider Trace

When you look for entries for the simpleform policy in the Embedded Service Provider trace, you can use the following strings to find the entries:

- The name of the Form Fill policy: `simpleform`
- The string identifying a Form Fill trace: `AGFormFill Policy Trace`
- The policy ID (after you have found it): `PolicyID#06OO287L-06LO-KKP4-207M-6971PPM6147L`

The following trace is from the `catalina.out` file of the Embedded Service Provider of an Access Gateway Appliance. The entries have been numbered so that they can be described, and a few extra line breaks and spaces have been added to make the entries easier to read.

```
1. <amLogEntry> 2009-09-14T00:15:52Z INFO NIDS Application: AM#501101050:
AMDEVICEID#esp-917A1174C8A270FC: PolicyID#06OO287L-06LO-KKP4-207M-6971PPM6147L:
NXPESID#2663: Evaluating policy </amLogEntry>

2. <amLogEntry> 2009-09-14T00:15:52Z INFO NIDS Application: AM#501104050:
AMDEVICEID#esp-917A1174C8A270FC: PolicyID#06OO287L-06LO-KKP4-207M-6971PPM6147L:
NXPESID#2663: AGFormFill Policy Trace:
   ~~RL~1~~~~Rule Count: 1~~Success(67)
   ~~RU~RuleID_1189711482510~simpleform~DNF~~0:1~~Success(67)
   ~~PA~ActionID_1189711485006~~Added Form Selection Group~~~~Success
       (0)
   ~~PA~ActionID_1189711485006~~Added Fill Options Group~~~~Success(0)
   ~~PA~ActionID_1189711485006~~Added Submit Options Group~~~~Success
       (0)
   ~~PC~ActionID_1189711485006~~Document=(ou=xpemlPEP,ou=mastercdn,
     ou=ContentPublisherContainer,ou=Partition,ou=PartitionsContainer,
     ou=VCDN_Root,ou=accessManagerContainer,o=novell:romaContent
     CollectionXMLDoc),Policy=(simpleform),Rule=(1::RuleID_11897114
     82510),Action=(FormFill::ActionID_1189711485006)~~~~Success(0)
 </amLogEntry>

3. <amLogEntry> 2009-09-14T00:15:52Z INFO NIDS Application: AM#501101021:
AMDEVICEID#esp-917A1174C8A270FC: PolicyID#06OO287L-06LO-KKP4-207M-6971PPM6147L:
NXPESID#2663: Response sent: Status - success </amLogEntry>
```

1. The first log entry is the request to evaluate the policy. If this entry doesn't occur, make sure that the Form Fill policy is enabled for the protected resource.
2. The second entry is the actual policy trace. For a Form Fill policy, it is fairly basic information about the three types of actions in the policy: matching the form, filling in the field options, and adding the submit options. To determine what information was put in the options, you need to view the proxy service trace.
3. The third entry indicates the type of response that is returned from the evaluation. In this entry, success is returned.

Proxy Service Trace

When you look for entries in the proxy trace of the Access Gateway log, you can use the following strings to find the entries:

- The event code of a Form Fill event: `AM#504507000`
- The name of the Form Fill policy: `simpleform`
- The name of the form: `mylogin`
- The names of the fill option fields: `username, password, title`

The sample trace is from a `ics_dyn.log` file of a Access Gateway Appliance. Some of the lines are very long, and extra white space has been added to make them easier to read. The first occurrence of an item you can search for is displayed in a bold typeface.

```
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0: AMEVENTID#0:
**************************************************
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Name : (mastercdnsimpleform3310)
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Type : (FILL)
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: CGI Matching Criteria: ()
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Page Matching Criteria:
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Not Configured.
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Form Number : (-1)
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Form Name: (mylogin)
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Form Id: ()
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Login Fail Redirect: ()
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Login Fail Delete Rem: ()
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Error Redirect: ()
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Post options (silent = yes), (debug = no), (masked =
  no), (enabled = yes)
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: InsertText: ()
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: JavaScriptHandling:
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Not configured.
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Fill Option 0 : ( Name=username, Value=NEPXurn~
  3Anovell~3Acredentialprofile~3A2005-03~2Fcp~3ASecrets~2Fcp~3ASecret~
  2Fcp~3AEntry~40~40~40~40WSCQSSToken~40~40~40~40~2Fcp~3ASecrets~
  2Fcp~3ASecret~5Bcp~3AName~3D~22LDAPCredentials~22~5D~2Fcp~3AEntry~
  5Bcp~3AName~3D~22UserName~22~5D, DataConversion=None,
  valType=CREDENTIAL_PROFILE, inputType=TEXT, isDuplicate=false)
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
  AMEVENTID#0: Fill Option 1 : ( Name=password, Value=NEPXurn~
  3Anovell~3Acredentialprofile~3A2005-03~2Fcp~3ASecrets~2Fcp~
  3ASecret~2Fcp~3AEntry~40~40~40~40WSCQSSToken~40~40~40~40~2Fcp~
  3ASecrets~2Fcp~3ASecret~5Bcp~3AName~3D~22LDAPCredentials~22~5D
  ~2Fcp~3AEntry~5Bcp~3AName~3D~22UserPassword~22~5D,
  DataConversion=None, valType=CREDENTIAL_PROFILE, inputType=PASSWORD,
  isDuplicate=false)
Sep 19 09:04:50 jwilson : AM#504507000: AMDEVICEID#ag-: AMAUTHID#0:
   AMEVENTID#0: Fill Option 2 : ( Name=title, Value=NEPXurn~
   3Anovell~3Aldap~3A2006-02~2Fldap~3AUserAttribute~40~40~40~
   40WSCQLDAPToken~40~40~40~40~2FUserAttribute~5B~40ldap~
   3AtargetAttribute~3D~22title~22~5D, DataConversion=None,
   valType=LDAP_ATTRIBUTE, inputType=TEXT, isDuplicate=false)
```

On the Access Gateway Appliance, you can get more detailed information on the process that was used to fill the form when you turn on logging to the `lagsoapmessages` file.

7.3 Common Configuration Problems That Prevent a Policy from Being Applied as Expected

When you try to determine what is functioning incorrectly in a policy, you need to turn on policy tracing and understand the evaluation traces. See the following:

- Section 7.1, "Turning on Logging for Policy Evaluation," on page 161
- Section 7.2, "Understanding Policy Evaluation Traces," on page 162

The CO entry line of a policy trace identifies when a policy condition evaluates to False or True. The PA entry line indicates whether the Action was applied or ignored. If the results of the policy trace are not what you expected for the user, the next step is to determine why the policy isn't behaving the way you want it to. Check for the following problems:

- Section 7.3.1, "Enabling Roles for Authorization Policies," on page 180
- Section 7.3.2, "LDAP Attribute Condition," on page 181
- Section 7.3.3, "Result on Condition Error Value," on page 181
- Section 7.3.4, "An External Secret Store and Form Fill," on page 182

7.3.1 Enabling Roles for Authorization Policies

If you are using roles in your authorization policies, you need to make sure that the role is enabled for the Identity Server configuration. You can create roles and authorization policies independently of assigning them to protect a resource or to an Identity Server configuration.

If you have not enabled the role, users are not assigned the role when they log in, even when they meet all the criteria for the role.

- If the Authorization Policy is an Allow policy, the users might be denied access because they haven't been assigned the role.
- If the Authorization Policy is a Deny policy, the users might be allowed access because they haven't been assigned the role.

Whenever an Authorization Policy is not producing the expected results and the policy contains a role, the first troubleshooting step should always be to check whether the role has been enabled for the Identity Server configuration. Click *Access Manager* > *Identity Servers* > *Edit* > *Roles*. If the role is not enabled, the Identity Server cannot assign the role to the user.

The second step should be to ensure that the roles are transferred from for Identity Server to the Embedded Service Provider. Click *Access Manager* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*. The *Authentication Profile* needs to be enabled in order for Embedded Service Providers to evaluate roles in policies. This profile is enabled by default, but it can be disabled. When it is disabled, all devices assigned to use this Identity Server cluster configuration cannot determine which roles a user has been assigned, and the devices evaluate policies as if the user has no roles.

7.3.2 LDAP Attribute Condition

If you use an LDAP attribute as the condition for a Role policy or an Authorization policy and your users are not being assigned the role or are denied access to a resource, the most likely cause of the problem is the LDAP attribute name used in the policy. Some administration tools for the LDAP user stores display a UI name or an eDirectory™ name rather than the LDAP attribute name. Access Manager policies require the LDAP attribute name.

Use the following steps to identity whether the Access Manager policy has been configured for the LDAP attribute name, a UI name, or an eDirectory name:

1. Use an LDAP browser to view one of your users in your LDAP user store.

   You can download a Java-based tool from the Internet.
2. Verify the LDAP name of the attribute and that the user has the expected value.
3. In the Administration Console, click *Policies* > *Policies* > *[Name of Policy]* > *Rule Number*.
4. View the attribute name and value for the LDAP Attribute condition.
5. Verify the following:
   - The name of the attribute should match the name as displayed in the LDAP browser. The attribute name is not case sensitive, but it should not contain any spaces. If you need to modify the attribute used by the policy, click the attribute name, then select an attribute from the list or select *New LDAP Attribute* to add one.
   - The value can be case sensitive, depending upon how you have configured the *Mode* for the policy. If you have selected case sensitive for the *Mode*, make sure the case in the policy matches the case in the LDAP user store.
   - If the attribute is multi-valued and your users typically have multiple values, select *Substring* as the *Comparison* type.
6. If these steps have not solved the problem, see Section 7.3.3, "Result on Condition Error Value," on page 181.

7.3.3 Result on Condition Error Value

If you incorrectly set the value of the *Result on Condition Error* field, you create a policy that allows an action that you want the policy to deny or that denies an action that you want allowed. You must carefully evaluate whether you want the action applied or ignored when an error occurs during the evaluation of the condition. For positive conditions, the following rules apply:

- For the action to be applied, either the user must match the condition or the *Result on Condition Error* must be set to True.
- For the action to be ignored, either the user must not match the condition or the *Result on Condition Error* must be set to False.

The logic is harder to follow when you start adding "if not" to the conditions. The user then matches the condition by not matching the condition. For this type of condition, you need to ask whether you want the action applied to any user when an error occurs evaluating the condition.

The logic is even harder to following when you start adding multiple condition groups that can also have "or nots" and "if nots".

If you have a policy that uses "if not" conditions or uses multiple condition groups and it is not producing the expected results, you might want to rewrite the policy so that it contains only positive conditions.

You might want to modify the condition groups so that the policy uses multiple rules, with each rule containing one condition group with the conditions you want the user to match for the action you assign to the rule.

7.3.4 An External Secret Store and Form Fill

When you create a user store on the Identity Server (*Local > User Stores*) and define it as an external Secret Store (*Liberty > Web Service Provider > Credential Profile*), some attributes are not being created properly on the SAML affiliate object. The workaround is to access the user store configuration page (*Local > User Stores*), then exit. This action results in a check to verify that the schema, objects, and attributes exist, and the affiliate object is then re-created from scratch, if necessary.

The following affiliate objects must exist:

```
authsamlCertContainerDN (container holding trusted certificates,
   for example: SCC Trusted Root.Security)
authsamlProviderID
authsamlTrustedCertDN (list of trusted certificate(s))
authsamlValidAfter (180 seconds default)
authsamlValidBefore (180 seconds default)
```

If these attributes exist, the system works normally. However, if your Identity Server and Secret Store server are not configured to use the same NTP server, time synchronization can be a problem. If time synchronization is an issue, you can change the 180-second default validity times as a workaround.

If your LDAP user store and the Administration Console have a firewall separating them, TCP ports 524 and 636 must be open to allow for the creation of the required objects. For more information about ports and firewalls, see "Setting Up Firewalls" in the *NetIQ Access Manager 3.2 Setup Guide*.

7.4 The Policy Is Using Old User Data

When a policy is first evaluated, it caches information about the user.

- Some data items are updated every minute.
- Some are cached for the duration of the request.
- Some are cached for the duration of the user's session. When a data item is cached for the duration of a user session, the user must log out and log in for the policy modification to take effect.

Table 7-8 lists how long the data items for a condition are cached before being refreshed.

Table 7-8 *Data Caching Limits*

Condition	Data Refresh Interval
Authenticating IDP	User session
Authentication Contract	User session
Authentication Method	User session
Authentication Type	User session
Client IP	Request
Credential Profile	User session

Condition	Data Refresh Interval
Current Date	One minute
Current Day of Week	One minute
Current Day of Month	One minute
Current Time of Day	One minute
HTTP Request Method	Request
Java Data Injection Module	User session
LDAP Attribute	User session; configurable to be cached only for the request with the Force Data Read option.
LDAP Group	User session
LDAP OU	User session
Liberty User Profile	User session
Proxy Session Cookie	User session
Roles for Current User	User session
Roles from Identity Provider	User session
Shared Secret	User session; configurable to be cached only for the request with the Force Data Read option.
String Constant	User session
URL	Request
URL Scheme	Request
URL Host	Request
URL Path	Request
URL File Name	Request
URL File Extension	Request
User Store	User session
X-Forward-For IP	Request

7.5 Form Fill and Identity Injection Silently Fail

Login with Form Fill or Identity Injection can fail when all of the following conditions occur:

- Your user store is configured to use Novell® SecretStore®.
- The shared secrets needed for Form Fill or Identity Injection are locked because the shared secrets are used by another application that is using the enhanced security feature. For example, if the application writes a secret called ssn, and you use that same secret in a Form Fill or Identity Injection policy, that secret is locked whenever the admin changes the user's password. Access Manager does not use the enhanced security feature when it writes shared secrets.

The new unlock feature for SecretStore can resolve this issue. See "Determining a Strategy for Unlocking the SecretStore" in the *NetIQ Access Manager 3.2 Identity Server Guide*.

7.6 Checking for Corrupted Policies

For a policy to be evaluated correctly, the policy must contain a rule. To verify that your system does not contain any policies with configuration errors:

1. In the Administration Console, click *Auditing* > *Troubleshooting* > *Policies*.

   If you have any corrupted policies, they appear in the list.
2. Identify the corrupted policy, then click *Remove*.

7.7 Policy Page Timeout

If your policy page hangs, and you have an LDAP group or LDAP ou being used in the policy, check the health of your user stores (LDAP servers) and ensure that they are communicating.

7.8 Policy Creation and Storage

For troubleshooting, you can export the policy and send it to Novell for debugging. If the policy uses roles, make sure you also export the Role policies.

Policies are stored as XML documents in the object directory, with one XML document to represent each policy container. The default policy container (Master_Container) resides at:

```
\\novell\accessManagerContainer\VCDN_Root\PartitionsContainer\Partition\ContentPub
lisherContainer\mastercdn\xpemlPEP\romaContentCollectionXMLDoc
```

Other policy containers are stored following the same path, with a unique name string representing the policy name that replaces the ou=mastercdn portion of the above path.

If you are unsure if the policy is being created correctly or if you need to check to see if the policy is enabled, you can view the policy list in the interface. If you think the GUI is not properly displaying the policy, you can also view the XML by navigating to the Policy Conditions on which you edit rules, right click and choose *This Frame* > *View Frame Source*.

7.9 Policy Distribution

Policy definitions are not replicated, but are referenced by the Access Gateways for which the policy is to be evaluated. The policy reference mechanism is a set of XML elements that refer back to the policy definitions stored in the various policy containers. If you have configured a policy for a protected resource and an Access Gateway does not seem to be executing this policy, use the following procedures to verify that the Access Gateway has been configured to use the policy:

1. Set the level of Application logging to *config*. See Section 7.1, "Turning on Logging for Policy Evaluation," on page 161.

   This enables the tracing of the policy enforcement lists.
2. Search for name of your policy in a `<PolicyEnforcementList>` element. The ExternalElementRef attribute contains a reference to the policy name.

   You can find these elements in the `catalina.out` file (Linux) or `stdout.log` file (Windows).

3 If you cannot find the policy name, the Access Gateway has not been configured to use the policy. The configuration either needs to be applied or the policy needs to be enabled. For information on how to assign a policy to a protected resource, see "Configuring Protected Resources" in the *NetIQ Access Manager 3.2 Access Gateway Guide*.

4 If you find the policy name associated with the correct protected resource, you need to check why the policy is not evaluating according to your design. Set the level of Application logging to *info* and examine the policy trace from a user accessing the protected resource. See Section 7.2, "Understanding Policy Evaluation Traces," on page 162.

7.10 Policy Evaluation: Access Gateway Devices

The following diagram depicts how Authorization policies fit into the protected resource processing for the proxy.

Figure 7-4 *Policy Evaluation*

Policies for the Access Gateway devices are evaluated by the policy engine in Java. A SOAP interface is used to transition from the proxy to Java and back. To see the SOAP messages, you need to set the logging level of the *Application* level to *config*. See Section 7.1, "Turning on Logging for Policy Evaluation," on page 161.

The SOAP messages are output to the `catalina.out` file (Linux) or `stdout.log` file (Windows). Sample SOAP messages are shown in the following scenarios:

- Section 7.10.1, "Successful Policy Configuration Example," on page 186
- Section 7.10.2, "No Policy Defined Configuration Example," on page 186
- Section 7.10.3, "Deny Access Configuration/Evaluation Example," on page 187

7.10.1 Successful Policy Configuration Example

Note the Policy Enforcement Point (PEP) identifier of AGIdentityInjection in the request and the PolicyID in the response.

Configuration Request

```
toBufSeg: <?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
   envelope/">
<SOAP-ENV:Body>
   <NXPES ID="12">
      <Configure-ag PEPName="AGIdentityInjection">
         <PolicyEnforcementList
            RuleCombiningAlgorithm="DenyOverridesWithPriority"
            schemaVersion="1.32"
            LastModified="1138389868885"
            LastModifiedBy="cn=admin,o=novell">
            <PolicyRef ElementRefType="ExternalWithIDRef"
                ExternalElementRef="PolicyID_xpemlPEP_AGIdentity
                    Injection_ii_test"
                ExternalDocRef="ou=xpemlPEP,ou=mastercdn,
                    ou=ContentPublisherContainer,ou=Partition,
                    ou=PartitionsContainer,ou=VCDN_Root,ou=access
                    ManagerContainer,o=novell:romaContentCollection
                    XMLDoc"
                UserInterfaceID="PolicyID_xpemlPEP_AGIdentity
                    Injection_ii_test"/>
         </PolicyEnforcementList>
      </Configure-ag>
   </NXPES>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Configuration Response

```
LibertyProcessMsgCB:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
   <NXPES Id="" Status="success">
      <ConfigureResponse PolicyId="755OK8P0-7543-518M-8L8M-N0P2LM2
                N3O27">
         <ContextDataElement Enum="2551"/>
      </ConfigureResponse>
   </NXPES>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

7.10.2 No Policy Defined Configuration Example

The following is a sample of a configuration request where the policy code detects that no policies are in effect for the protected resource and Policy Enforcement Point (PEP).

Configuration Request

```
toBufSeg: <?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
   <NXPES ID="11">
      <Configure-ag PEPName="AGAuthorization">
         <PolicyEnforcementList
            RuleCombiningAlgorithm="DenyOverridesWithPriority"
            schemaVersion="1.32"
            LastModified="1138389868885"
            LastModifiedBy="cn=admin,o=novell">
           <PolicyRef ElementRefType="ExternalWithIDRef"
               ExternalElementRef="PolicyID_xpemlPEP_AGIdentity
                        Injection_ii_test"
               ExternalDocRef="ou=xpemlPEP,ou=mastercdn,ou=Content
                       PublisherContainer,ou=Partition,ou=Partitions
                       Container,ou=VCDN_Root,ou=accessManager
                       Container,o=novell:romaContentCollectionXMLDoc"
               UserInterfaceID="PolicyID_xpemlPEP_AGIdentityInjection_
                       ii_test"/>
         </PolicyEnforcementList>
      </Configure-ag>
   </NXPES>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Configuration Response

```
LibertyProcessMsgCB:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
      envelope/">
   <SOAP-ENV:Body>
      <NXPES Id="" Status="emptypolicyset"/>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

7.10.3 Deny Access Configuration/Evaluation Example

The following is a sample of a configuration request for a Deny policy and an evaluation request for this policy.

Configuration Request

```
toBufSeg: <?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
   envelope/">
<SOAP-ENV:Body>
   <NXPES ID="17">
      <Configure-ag PEPName="AGAuthorization">
         <PolicyEnforcementList
            RuleCombiningAlgorithm="DenyOverridesWithPriority"
            schemaVersion="1.32"
            LastModified="1138718667305"
            LastModifiedBy="cn=admin,o=novell">
         <PolicyRef
            ElementRefType="ExternalWithIDRef"
            ExternalElementRef="PolicyID_xpemlPEP_AGIdentityInjection
                _custom_test"
            ExternalDocRef="ou=xpemlPEP,ou=mastercdn,ou=Content
               PublisherContainer,ou=Partition,ou=PartitionsContainer,
               ou=VCDN_Root,ou=accessManagerContainer,o=novell:roma
               ContentCollectionXMLDoc"
            UserInterfaceID="PolicyID_xpemlPEP_AGIdentityInjection
```

```
                _custom_test"/>
            <PolicyRef
               ElementRefType="ExternalWithIDRef"
               ExternalElementRef="PolicyID_xpemlPEP_AGAuthorization_
                  deny-all"
               ExternalDocRef="ou=xpemlPEP,ou=mastercdn,ou=Content
                  PublisherContainer,ou=Partition,ou=PartitionsContainer,
                  ou=VCDN_Root,ou=accessManagerContainer,o=novell:roma
                  ContentCollectionXMLDoc"
               UserInterfaceID="PolicyID_xpemlPEP_AGAuthorization
                  _deny-all"/>
            </PolicyEnforcementList>
         </Configure-ag>
      </NXPES>
   </SOAP-ENV:Body>
   </SOAP-ENV:Envelope>
```

Configuration Response

```
LibertyProcessMsgCB:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
    envelope/">
<SOAP-ENV:Body>
   <NXPES Id="" Status="success">
      <ConfigureResponse
           PolicyId="55N3NL81-L29N-2619-K0M8-2L963M0MM701"/>
   </NXPES>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Evaluation Request

```
toBufSeg: <?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
   <NXPES ID="18">
      <Evaluate PolicyId="55N3NL81-L29N-2619-K0M8-2L963M0MM701"
                Verbose="on"/>
   </NXPES>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Evaluation Response

```
LibertyProcessMsgCB:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
      envelope/">
<SOAP-ENV:Body>
   <NXPES Id="" Status="success">
      <EvaluateResponse>
         <DoAction ActionName="Deny" ActionTTL="-1" Enum="2620">
            <Parameter Enum="10" Name="Message" Value=""/>
         </DoAction>
      </EvaluateResponse>
   </NXPES>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```