AUTHORIZED DOCUMENTATION

Identity Server Guide

Novell®
Access Manager

3.1 SP2

November 16, 2010

www.novell.com

Legal Notices

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

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

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

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

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

Novell Trademarks

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

Third-Party Materials

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

Contents

About This Guide

This guide describes the following features of the Identity Server:

This guide is intended to help you understand and configure all of the features provided by the Identity Server.

It is recommended that you first become familiar with the information in the *Novell Access Manager 3.1 SP2 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.

The setup guide and this guide are designed to work together, and important information and setup steps are not always repeated in both places.

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, or go to Documentation Feedback (http://www.novell.com/documentation/feedback.html) at www.novell.com/documentation/feedback.html and enter your comments there.

Documentation Updates

For the most recent version of the *Access Manager Identity Server Guide*, visit the Novell Access Manager Documentation Web site (http://www.novell.com/documentation/novellaccessmanager31/).

Additional Documentation

Before proceeding, you should be familiar with the *Novell Access Manager 3.1 SP2 Installation Guide* and the *Novell Access Manager 3.1 SP2 Setup Guide*, which provide information about installing and setting up the Access Manager system.

If you are unfamiliar with SAML 1.1, see "SAML Overview" (http://www.novell.com/documentation/saml/saml/data/ag8qdk7.html) on the Documentation Web site (http://www.novell.com/documentation/a-z.html).

For conceptual information about Liberty, and to learn about what is new for SAML 2.0, see Appendix A, "About Liberty," on page 367 and Appendix B, "Understanding How Access Manager Uses SAML," on page 369.

For information about the other Access Manager devices and features, see the following:

- *Novell Access Manager 3.1 SP2 Administration Console Guide*
- *Novell Access Manager 3.1 SP2 Access Gateway Guide*
- *Novell Access Manager 3.1 SP2 Policy Guide*
- *Novell Access Manager 3.1 SP2 J2EE Agent Guide*
- *Novell Access Manager 3.1 SP2 SSL VPN Server Guide*
- *Novell Access Manager 3.1 SP2 Event Codes*

Configuring an Identity Server

1

After you log in to the Administration Console, click *Devices* > *Identity Servers*. The system displays the Identity Servers that can be managed from this Administration Console.

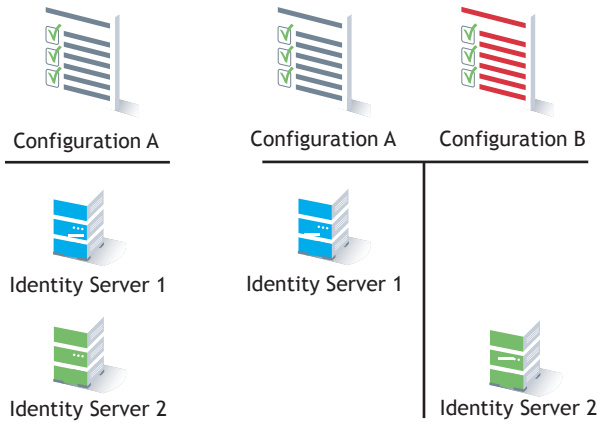
A newly installed Identity Server is in an unconfigured state and is halted. It remains in this state and cannot function until you create a cluster configuration and assign the Identity Server to the new configuration. The cluster configuration defines how the Identity Server functions in an Access Manager configuration. You can assign multiple servers to use the same configuration, which enables failover and load balancing services.

For information on configuring local authentication options, see the following:

1.1 Managing a Cluster Configuration

After you install an Identity Server, you must create a cluster configuration in order to configure the Identity Server. Even if you have only one Identity Server, you must assign it to a cluster configuration to configure it. If you have multiple Identity Servers, you can create multiple configurations and assign different Identity Servers to them as shown in Figure 1-1.

Figure 1-1 *Identity Server Configurations*



When you assign multiple Identity Servers to the same configuration, you need to install a load balancer that supports either Layer 4 or Layer 7. This device is referred to as an L4 switch in this manual. The L4 switch allows the work load to be balanced among the machines.

Whether you have one machine or multiple machines in a cluster, the Access Manager software configuration process is the same. This section describes the following cluster management tasks:

- Section 1.1.1, “Creating a Cluster Configuration,” on page 16
- Section 1.1.2, “Assigning an Identity Server to a Cluster Configuration,” on page 21
- Section 1.1.3, “Configuring a Cluster with Multiple Identity Servers,” on page 21
- Section 1.1.4, “Configuring Session Failover,” on page 22
- Section 1.1.5, “Editing Cluster Details,” on page 24
- Section 1.1.6, “Removing a Server from a Cluster Configuration,” on page 25
- Section 1.1.7, “Enabling and Disabling Protocols,” on page 25
- Section 1.1.8, “Modifying the Base URL,” on page 26

1.1.1 Creating a Cluster Configuration

This section discusses all the settings available when creating an Identity Server configuration. If you want a description of just the options required to get a functional configuration, see “Creating a Basic Identity Server Configuration” in the *Novell Access Manager 3.1 SP2 Setup Guide*.

An Identity Server always operates as an identity provider and can optionally be configured to run as an identity consumer (also known as a service provider), using Liberty, SAML 1.1, SAML 2.0, CardSpace or WS Federation protocols. These topics are not described in this section.

In an Identity Server configuration, you specify the following information:

- The DNS name for the Identity Server or clustered server site.
- Certificates for the Identity Server.

- Organizational and contact information for the server, which is published in the metadata of the Liberty and SAML protocols.
- The LDAP directories (user stores) used to authenticate users, and the trusted root for secure communication between the Identity Server and the user store.

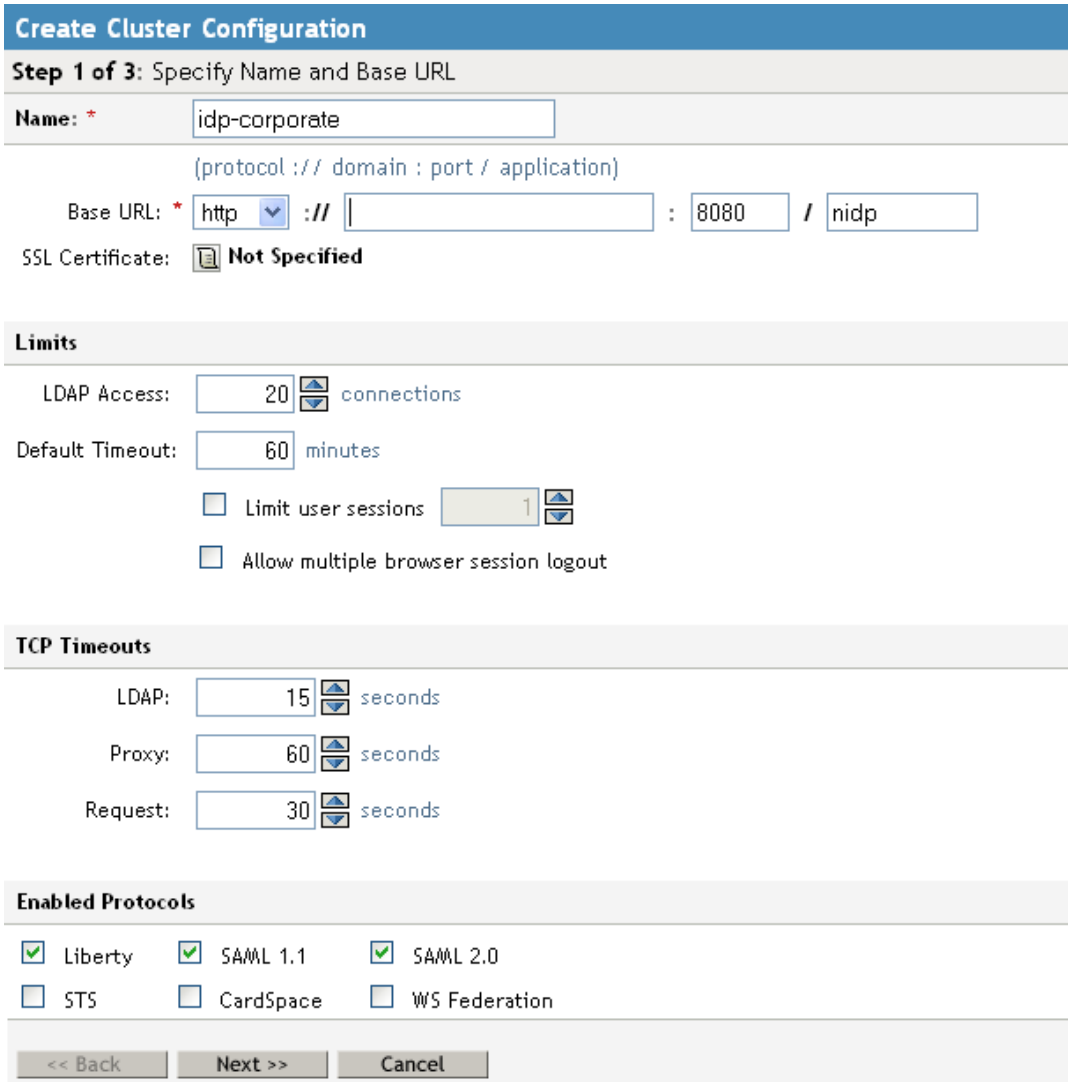
To create an Identity Server configuration:

1. In the Administration Console, click *Devices* > *Identity Servers*.
2. Select the Identity Server's check box, then click *New Cluster*.

   Selecting the server is one way to assign it to the cluster configuration.
3. In the *New Cluster* dialog box, specify a name for the cluster configuration.

   If you did not select the server in the previous step, you can now select the server or servers that you want to assign to this configuration.

   For more information about assigning servers to a configuration, see Section 1.1.2, "Assigning an Identity Server to a Cluster Configuration," on page 21.
4. Click *OK*.



5 Fill in the following fields to specify the Base URL for your Identity Server configuration:

Name: Specify a name by which you want to refer to the configuration. This field is populated with the name you provided in the *New Cluster* dialog box. You can change this name here, if necessary.

IMPORTANT: Carefully determine your settings for the base URL, protocol, and domain. After you have configured trust relationships between providers, changing these settings invalidates the trust model and requires a reimport of the provider's metadata.

Modifying the base URL also invalidates the trust between the Embedded Service Provider of Access Manager devices. To re-establish the trust after modifying the base URL, you must restart the Embedded Service Provider on each device.

Base URL: Specify the application path for the Identity Server. The Identity Server protocols rely on this base URL to generate URL endpoints for each protocol.

- **Protocol:** Select the communication protocol. Specify HTTPS in order to run securely (in SSL mode) and for provisioning. Use HTTP only if you do not require security or have installed an SSL terminator in front of the Identity Server.
- **Domain:** Specify the DNS name assigned to the Identity Server. When you are using an L4 switch, this DNS name should resolve to the virtual IP address set up on the L4 switch for the Identity Servers. Using an IP address is not recommended.
- **Port:** Specify the port value for the protocol. Default ports are 8080 for HTTP or 8443 for HTTPS. If you want to use port 80 or 443, specify the port here.
  - If you are configuring a Linux Identity Server, you must also configure the operating system to translate the port. See Section 1.5, "Translating the Identity Server Configuration Port," on page 36.
  - If you are configuring a Windows Identity Server, you must also modify the Tomcat `server.xml` file located in the `\Program Files\Novell\Tomcat\conf` directory for Windows Server 2003 or the `\Program Files (x86)\Novell\Tomcat\conf` directory for Windows Server 2008. Change the ports from 8080 and 8443 to 80 and 443, then restart the Tomcat service.
- **Application:** Specify the Identity Server application. Leave the default value *nidp*.

SSL Certificate: Displays the currently assigned SSL certificate.

The Identity Server comes with a `test-connector` certificate that you must replace to use SSL in your production environment. You can replace the test certificate now or after you configure the Identity Server. If you create the certificate and replace the `test-connector` certificate now, you can save some time by restarting Tomcat only once. Tomcat must be restarted whenever you assign an Identity Server to a configuration and whenever you update a certificate key store. See Section 1.3.3, "Managing the Keys, Certificates, and Trust Stores," on page 29.

For information on how to replace the `test-connector` certificate, see "Enabling SSL Communication" in the *Novell Access Manager 3.1 SP2 Setup Guide*.

6 To configure session limits, fill in the following fields:

LDAP Access: Specify the maximum number of LDAP connections the Identity Server can create to access the configuration store. You can adjust this amount for system performance.

Default Timeout: Specify the session timeout you want assigned as a default value when you create a contract. This value is also assigned to a session when the Identity Server cannot associate a contract with the authenticated session. During federation, if the authentication request uses a type rather than a contract, the Identity Server cannot always associate a contract with the request.

The traditional SSL VPN server uses the *Any Contract* option for authentication. The user is assigned the timeout value of the contract used for authentication, and not this default timeout value.

If you want to know what timeout value the SSL VPN user is assigned, you need to select a contract with the appropriate timeout value. Click *Devices > Access Gateways > [Name of Reverse Proxy] > [Name of Proxy Service] > SSLVPN_Default*. The *SSLVPN_Default* name is the default name for the SSL VPN protected resource. If you have modified this name, select that protected resource. In the Authentication Procedure option, select a name/password contract with the appropriate timeout value.

Limit User Sessions: Specify whether user sessions are limited. If selected, you can specify the maximum number of concurrent sessions a user is allowed to authenticate.

If you decide to limit user sessions, you should also give close consideration to the session timeout value (the default is 60 minutes). If the user closes the browser without logging out (or an error causes the browser to close), the session is not cleared until the session timeout expires. If the user session limit is reached and those sessions have not been cleared with a logout, the user cannot log in again until the session timeout expires for one of the sessions.

When enabled, this option affects performance in a cluster with multiple Identity Servers. When a user is limited to a specific number of sessions, the Identity Servers must check with the other servers before establishing a new session.

Allow multiple browser session logout: Specify whether a user with more than one session to the server is presented with an option to log out of all sessions. If you do not select this option, only the current session can be logged out. Deselect this option in instances where multiple users log in as guests. Then, when one user logs out, none of the other guests are logged out.

When you enable this option, you must also restart any Embedded Service Providers that use this Identity Server configuration.

7 To configure TCP timeouts, fill in the following fields:

LDAP: Specify how long an LDAP request to the user store can take before timing out.

Proxy: Specify how long a request to another cluster member can take before timing out. When a member of a cluster receives a request from a user who has authenticated with another cluster member, the member sends a request to the authenticating member for information about the user.

Request: Specify how long an HTTP request to another device can take before timing out.

8 To control which protocols can be used for authentication, select one or more of the following protocols.

IMPORTANT: Enable only the protocols that you are using.

If you are using other Access Manager devices such as the Access Gateway, SSL VPN, or the J2EE Agents, you need to enable the Liberty protocol. The Access Manager devices use an Embedded Service Provider. If you disable the Liberty protocol, you disable the trusted relationships these devices have with the Identity Server, and authentication fails.

Liberty: Uses a structured version of SAML to exchange authentication and authorization data between trusted identity providers and service providers and provides the framework for user federation.

SAML 1.1: Uses XML for exchanging authentication and authorization data between trusted identity providers and service providers.

SAML 2.0: Uses XML for exchanging encrypted authentication and authorization data between trusted identity providers and service providers and provides the framework for user federation.

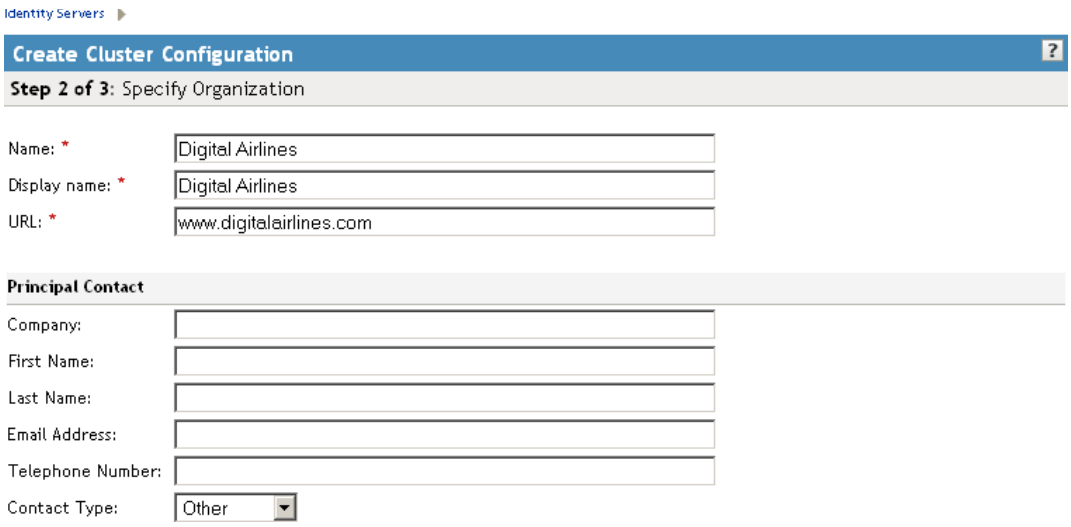
STS: A security token service that creates digital identities from claims, which can then be used as a card or a token for authentication.

CardSpace: Uses Microsoft client software that stores a user's information in a digital identity or information card, which can then be presented and used as authentication credentials.

WS Federation: Allows disparate security mechanisms to exchange information about identities, attributes, and authentication.

9 To continue creating the Identity Server configuration, click *Next*.

The system displays the Organization page.



Use this page to specify organization information for the Identity Server configuration. The information you specify on this page is published in the metadata for the Liberty 1.2 and SAML protocols. The metadata is traded with federation partners and supplies various information regarding contact and organization information located at the Identity Server.

The following fields require information:

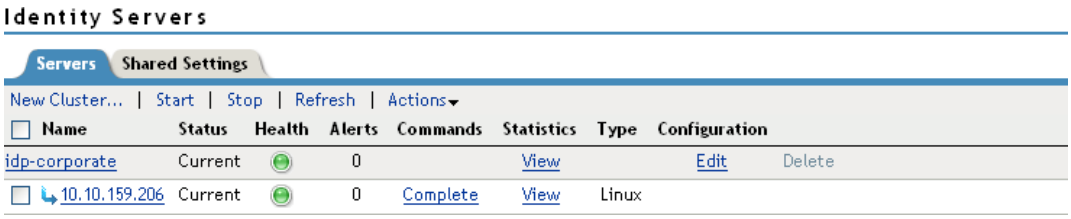
- **Name:** The name of the organization.
- **Display Name:** The display name for the organization.
- **URL:** The organization's URL for contact purposes.

Optional fields include Company, First Name, Last Name, Email, Telephone, and Contact Type.

10 Click *Next* to configure the user store.

You must reference your own user store and auto-import the SSL certificate. See Section 3.1, "Configuring Identity User Stores," on page 104 for information about this procedure.

11 After you configure the user store, the system displays the new configuration on the Servers page.



The status icons for the configuration and the Identity Server should turn green. It might take several seconds for the Identity Server to start and for the system to display a green icon. If it does not, it is likely that the Identity Server is not communicating with the user store you set up. Ensure that you have entered the user store information correctly, and that you imported the SSL certificate to the user store. (*Edit* > *Local* > *[User Store Name]*.)

1.1.2 Assigning an Identity Server to a Cluster Configuration

After you create a configuration, you must assign an Identity Server to it. For clustering, you can assign more than one Identity Server to the configuration (see Section 1.1.3, "Configuring a Cluster with Multiple Identity Servers," on page 21 for the steps to set up a cluster). A configuration uses any shared settings you have specified, such as attribute sets, user matching expressions, and custom attributes that are defined for the server.

1 In the Administration Console, click *Devices* > *Identity Servers*.

2 On the Servers page, select the server's check box.

You can select all displayed servers by selecting the top-level Server check box.

3 Click *Actions* > *Assign to Cluster*.

4 Select the configuration's check box, then click *Assign*.

You are prompted to restart Tomcat. The status icon for the Identity Server should turn green. It might take several seconds for the Identity Server to start and for the system to display the green icon.

1.1.3 Configuring a Cluster with Multiple Identity Servers

To add capacity and to enable system failover, you can cluster a group of Identity Servers and configure them in a cluster configuration to act as a single server. You can also configure the cluster to support session failover, so that users don't have to reauthenticate when an Identity Server goes down.

A cluster of Identity Servers should reside behind an L4 switch. Clients access the virtual IP (VIP) address of the cluster presented on the L4 switch, and the L4 switch alleviates server load by balancing traffic across the cluster. Whenever a user accesses the virtual IP address assigned to the L4 switch, the system routes the user to one of the Identity Servers in the cluster, as traffic necessitates.

To set up a cluster, complete the following tasks:

- ❑ Install an L4 switch. You can use the same switch for Identity Server clustering and Access Gateway clustering, provided that you use different virtual IPs. The LB algorithm can be anything (hash/sticky bit), defined at the Real server level. For configuration tips, see "Configuration Tips for the L4 Switch " in the *Novell Access Manager 3.1 SP2 Setup Guide*.
- ❑ Enable persistence (sticky) sessions on the L4 switch. Normally you define this at the virtual server level.
- ❑ Create an Identity Server configuration for the cluster. You assign all the Identity Servers to this configuration.
  - See Section 1.1.1, "Creating a Cluster Configuration," on page 16 for information about creating an Identity Server configuration.
  - See Section 1.1.2, "Assigning an Identity Server to a Cluster Configuration," on page 21 for information about assigning identity servers to configurations.
- ❑ Ensure that the DNS name of the base URL for the cluster configuration resolves via DNS to the IP address of the L4 virtual IP address. The L4 switch balances the load between the Identity Servers in the cluster.
- ❑ Ensure that the L4 administration server using port 8080 has the following TCP ports open:
  - 8443 (secure Administration Console)
  - 7801 + 1 (for back-channel communication with cluster members). You need to open two consecutive ports, such as 7801 and 7802.
  - 636 (for secure LDAP)
  - 389 (for clear LDAP)
  - 524 (network control protocol on the L4 switch for server communication)

  The identity provider ports must also be open:
  - 8080 (non-secure login)
  - 8443 (secure login)
  - 1443 (server communication)
- ❑ If you are using introductions (see Section 7.2, "Configuring General Provider Options," on page 190), you must configure the L4 switch to load balance on ports 8445 (identity provider) and 8446 (identity consumer).
- ❑ To enable session failover so users don't have to reauthenticate when an Identity Server goes down, see Section 1.1.4, "Configuring Session Failover," on page 22.
- ❑ To modify the name of the cluster or edit communication details, see Section 1.1.5, "Editing Cluster Details," on page 24.

1.1.4 Configuring Session Failover

When you set up an Identity Server cluster and add more than one Identity Server to the cluster, you have set up fault tolerance. This ensures that if one of the Identity Servers goes down, users still have access to your site because the remaining Identity Server can be used for authentication. However, it doesn't provide session failover. If a user has authenticated to the failed Identity Server, that user is prompted to authenticate and the session information is lost.

When you enable session failover and an Identity Server goes down, the user's session information is preserved. Another peer server in the cluster re-creates the authoritative session information in the background. The user is not required to log in again and experiences no interruption of services.

Prerequisites

- An Identity Server cluster with two or more Identity Servers.
- Sufficient memory on the Identity Servers to store additional authentication information. When an Identity Server is selected to be a failover peer, the Identity Server stores about 1 KB of session information for each user authenticated on the other machine.
- Sufficient network bandwidth for the increased login traffic. The Identity Server sends the session information to all the Identity Servers that have been selected to be its failover peers.
- All trusted Embedded Services Providers need to be configured to send the attributes used in Form Fill and Identity Injection policies at authentication. If you use any attributes other than the standard credential attributes in your contracts, you also need to send these attributes. To configure the attributes to send, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *[Name of Service Provider]* > *Attributes*.

Configuring Session Failover

1. In the Administration Console, click *Devices* > *Identity Servers*.
2. In the list of clusters and Identity Servers, click the name of an Identity Server cluster.
3. Click the *IDP Failover Peer Server Count*, then select the number of failover peers you want each Identity Server to have.
   - To disable this feature, select 0.
   - To enable this feature, select one or two less than the number of servers in your cluster. For example, if you have 4 servers in your clusters and you want to allow for one server being down for maintenance, select 3 (4-1=3). If you want to allow for the possibility of two servers being down, select 2 (4-2=2).

     If you have eight or more servers in your cluster, the formula 8-2=6 gives each server 6 peers. This is probably more peers than you need for session failover. In a larger cluster, you should probably limit the number of peers to 2 or 3. If you select too many peers, your machines might require more memory to hold the session data and you might slow down your network with the additional traffic for session information.
4. Click *OK*.

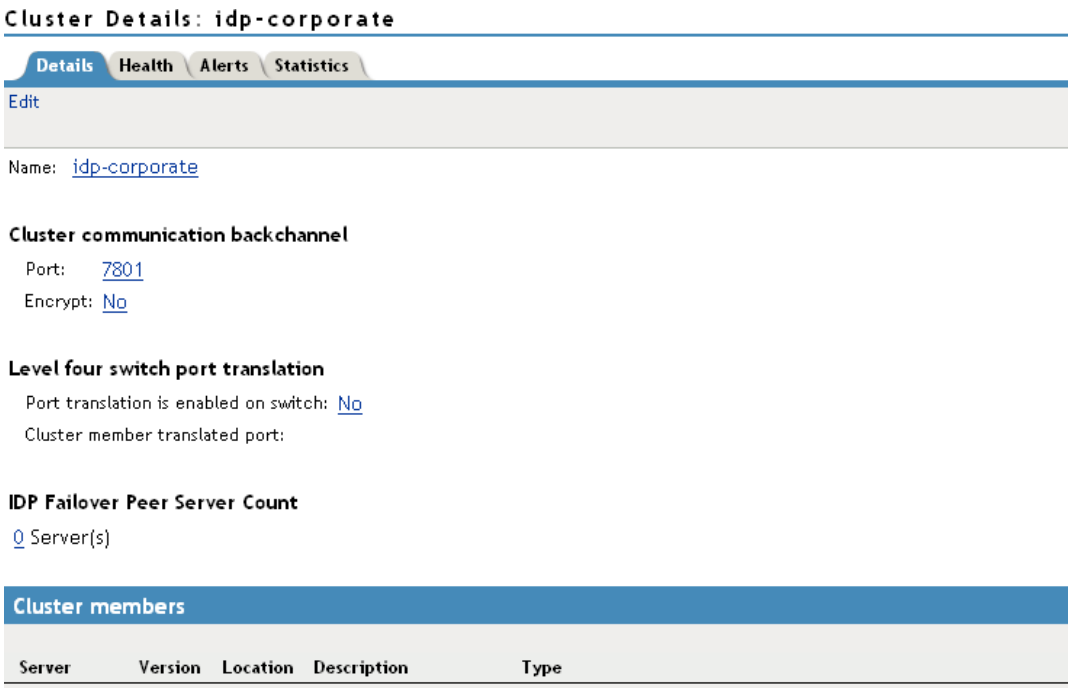
How Failover Peers Are Selected

The failover peers for an Identity Server are selected according to their proximity. Access Manager sorts the members of the cluster by their IP addresses and ranks them according to how close their IP addresses are to the server who needs to be assigned failover peers. It selects the closest peers for the assignment. For example, if a cluster member exists on the same subnet, that member is selected to be a failover peer before a peer that exists on a different subnet.

1.1.5 Editing Cluster Details

The Cluster Details page lets you manage the configuration's cluster details, health, alerts, and statistics.

1. In the Administration Console, click *Devices* > *Identity Servers*.
2. Click the name of the cluster configuration.



3. Select from the following actions:

   Details: To modify the cluster name or its settings, click *Edit*, then continue with Step 4.

   Health: To view the health of the cluster, click the *Health* tab.

   Alerts: To view the alerts generated by members of the cluster, click the *Alerts* tab.

   Statistics: To view the statistics of the cluster members, click the *Statistics* tab.

4. Modify the following fields as required:

   Cluster Communication Backchannel: Specify a communications channel over which the cluster members maintain the integrity of the cluster. For example, this TCP channel is used to detect new cluster members as they join the cluster, and to detect members that leave the cluster. A small percentage of this TCP traffic is used to help cluster members determine which cluster member would best handle a given request. This back channel should not be confused with the IP address/port over which cluster members provide proxy requests to peer cluster members.

   - **Port:** Specify the TCP port of the cluster back channel on all of the Identity Servers in the cluster. 7801 is the default TCP port.

     Because the cluster back channel uses TCP, you can have cluster members on different networks. However, firewalls must allow the ports specified here plus one to pass through. You need to open two ports for each cluster, for example, 7801 and 7802.

   - **Encrypt:** Encrypts the content of the messages that are sent between cluster members.

Level Four Switch Port Translation: Configure the L4 switch to translate the port of the incoming request to a new port when the request is sent to a cluster member. Because the cluster members communicate with each other over the same IP address/port as the L4 switch, the cluster implementation needs to know what that port is. The translated port is the port on the cluster members where other cluster members can contact it. This is the IP address and port where cluster members provide proxy requests to other cluster members.

- **Port translation is enabled on switch:** Specify whether the port of the L4 switch is different from the port of the cluster member. For example, enable this option when the L4 switch is using port 443 and the Identity Server is using port 8443.
- **Cluster member translated port:** Specify the port of the cluster member.

IDP Failover Peer Server Count: For configuration information, see Section 1.1.4, "Configuring Session Failover," on page 22.

5 Click *OK*, then update the Identity Server as prompted.

1.1.6 Removing a Server from a Cluster Configuration

Removing an Identity Server from a configuration disassociates the Identity Server from the cluster configuration. The configuration, however, remains intact and can be reassigned later or assigned to another server.

1 In the Administration Console, click *Devices* > *Identity Servers*.

2 Select the server, then click *Stop*. Wait for the Health indicator to turn red.

3 Select the server, then choose *Actions* > *Remove from Cluster*.

For information about deleting an Identity Server, see Section 14.1, "Managing an Identity Server," on page 321.

1.1.7 Enabling and Disabling Protocols

You can control which protocols can be used for authenticating with an Identity Server configuration. A protocol must be enabled and configured before users can use the protocol for authentication. For tight security, consider disabling the protocols that you are not going to use for authentication.

When you disable a protocol, updating the Identity Server configuration is not enough. You must stop and start the Identity Server.

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

2 In the *Enabled Protocols* section, select the protocols to enable

3 To disable a protocol, deselect it.

4 Click *OK*.

5 (Conditional) If you have enabled a protocol, update the Identity Server.

6 (Conditional) If you have disabled a protocol, stop and start the Identity Server.

6a Select the Identity Server, then click *Stop*.

6b When the health turns red, select the Identity Server, then click *Start*.

6c Repeat the process for each Identity Server in the cluster.

1.1.8 Modifying the Base URL

When you configure an Identity Server, you must carefully determine your settings for the base URL, protocol, and domain. Changing the base URL invalidates the trust model and requires a reimport of the provider's metadata, and a restart of the affected Embedded Service Providers. It also changes the ID of the provider and the URLs that others use for access.

When you change the base URL of the Identity Server, you invalidate the following trusted relationships:

- The trusted relationships that the Identity Server has established with each Access Manager device that has been configured to use the Identity Server for authentication
- The trusted relationship that each Access Manager device has established with the Identity Server when the Identity Server configuration was selected.
- The trusted relationships that the Identity Server has established with other service providers.

The sessions of any logged-in users are destroyed and no user can log in and access protected resources until the trust relationships are reestablished.

To modify the base URL and reestablish trust relationships:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit*.
2. Change the protocol, domain, port, and application settings, as necessary.
3. Click *OK*.
4. On the Identity Servers page, click *Update*.

   This re-creates the trusted Identity Server configuration to use the new Base URL and metadata.
5. Restart Tomcat on each Identity Server in the configuration:
   - **Linux Identity Server:** Enter the following command:

     ```
     /etc/init.d/novell-tomcat5 restart
     ```
   - **Windows Identity Server:** Enter the following commands:

     ```
     net stop Tomcat5
     net start Tomcat5
     ```
6. For each Access Manager device configured to trust the configuration of this modified base URL, you must update the device so that the Embedded Service Provider trusts the new Identity Server configuration:
   - Click *Access Gateways*, then click *Update* for any servers with a *Status* of *Update*.
   - Click *SSL VPNs*, then click *Update* for any servers with a *Status* of *Update*.
   - Click *J2EE Agents*, then click *Update* for any agents with a *Status* of *Update*.
7. For each service provider you have configured to trust the configuration of this modified base URL, you must send them the new metadata and have them re-import it.

For information about setting up SSL and changing an Identity Server from HTTP to HTTPS, see "Enabling SSL Communication" in the *Novell Access Manager 3.1 SP2 Setup Guide*.

1.2 Enabling Role-Based Access Control

Role-based access control is used to provide a convenient way to assign a user to a particular job function or set of permissions within an enterprise, in order to control access. In Access Manager, you assign users to roles, based on attributes of their identity, and then associate authorization policies to the role.

For a complete discussion on creating and configuring role policies, see "Creating Role Policies" in the *Novell Access Manager 3.1 SP2 Policy Guide*.

In order for a role to be assigned to users at authentication, you must enable it for the Identity Server configuration.

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

1.3 Configuring Secure Communication on the Identity Server

The Identity Server uses the following key pairs for secure communication. In a production environment, you should exchange the key pairs that are created at installation time with certificates from a trusted certificate authority.

- **Connector:** The `test-connector` certificate is used when you establish SSL communication between the Identity Server and the browsers and between the Identity Server and the Access Gateway for back-channel communications. It needs to be replaced with a certificate that has a subject name that matches the DNS name of the Identity Server. This task is part of basic setup. See "Enabling SSL Communication" in the *Novell Access Manager 3.1 SP2 Setup Guide*.
- **Signing:** The test-signing key pair is used by the various protocols to sign authentication requests, to sign communication with providers on the SOAP back channel, and to sign Web Service Provider profiles. For more information on the services that use the signing certificate, see "Access Manager Services That Use the Signing Certificate" on page 42.

  This certificate can be stored in an external HSM keystore. For information on how to use netHSM to replace and manage this signing certificate, see Section 1.6, "Using netHSM for the Signing Key Pair," on page 41.
- **Data Encryption:** The test-encryption key pair is used to encrypt specific fields or data in the assertions. For more information on the services that use the encryption certificate, see Section 1.3.2, "Viewing Services That Use the Encryption Key Pair," on page 29.

To force the browser connections to the Identity Server to support a specific level of encryption, see Section 1.4.3, "Forcing 128-Bit Encryption," on page 33.

If you are going to use introductions in your federation configuration, you need to set up the following key pairs:

- **Identity provider:** The test-provider key pair is used when you configure your Identity Server to use introductions with other identity providers and have set up a common domain name for this purpose. It needs to be replaced with a certificate that has a subject name that matches the DNS name of the common domain. For configuration information, see Section 7.2.1, "Configuring the General Identity Provider Options," on page 190.
- **Identity consumer:** The test-consumer key pair is used when you configure your Identity Server to use introductions with other service providers and have set up a common domain name for this purpose. It needs to be replaced with a certificate that has a subject name that matches the DNS name of the common domain. For configuration information, see Section 7.2.2, "Configuring the General Identity Consumer Options," on page 191.

To enable secure communication between the user store and the Identity Server, you can also import the trusted root certificate of the user store. For configuration information, see Section 3.1, "Configuring Identity User Stores," on page 104.

This section describes the following tasks:

1.3.1 Viewing the Services That Use the Signing Key Pair

The following services can be configured to use signing:

Protocols

The protocols can be configured to sign authentication requests and responses.

To view your current configuration:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit*.
2. In the *Identity Provider* section, view the setting for the *Require Signed Authentication Requests* option. If it is selected, all authentication requests from identity providers are signed.
3. In the *Identity Consumer* section, view the settings for the *Require Signed Assertion*s and S*ign Authentication Requests* options. If these options are selected, assertions and authentication requests are signed.

SOAP Back Channel

The SOAP back channel is the channel that the protocols use to communicate directly with a provider. The SOAP back channel is used for artifact resolutions and attribute queries for the Identity Web Services Framework.

To view your current configuration for the SOAP back channel:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit*.
2. Select the protocol (Liberty, SAML 1.1, or SAML 2.0), then click the name of an identity provider or service provider.
3. Click *Access*.
4. View the *Security* section. If the *Message Signing* option is selected, signing is enabled for the SOAP back channel.

Profiles

Any of the Web Service Provider profiles can be enabled for signing by configuring them to use X.509 for their message-level security mechanism.

To view your current configuration:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.
2. Click the name of a profile, then click *Descriptions*.
3. Click the *Description Name*.
4. If either *Peer entity = None, Message=X509* or *Peer entity = MutualTLS, Message=X509* has been selected as the security mechanism, signing has been enabled for the profile.

1.3.2 Viewing Services That Use the Encryption Key Pair

All of the Liberty Web Service Provider Profiles allow you to configure them so that the resource IDs are encrypted. By default, no profile encrypts the IDs.

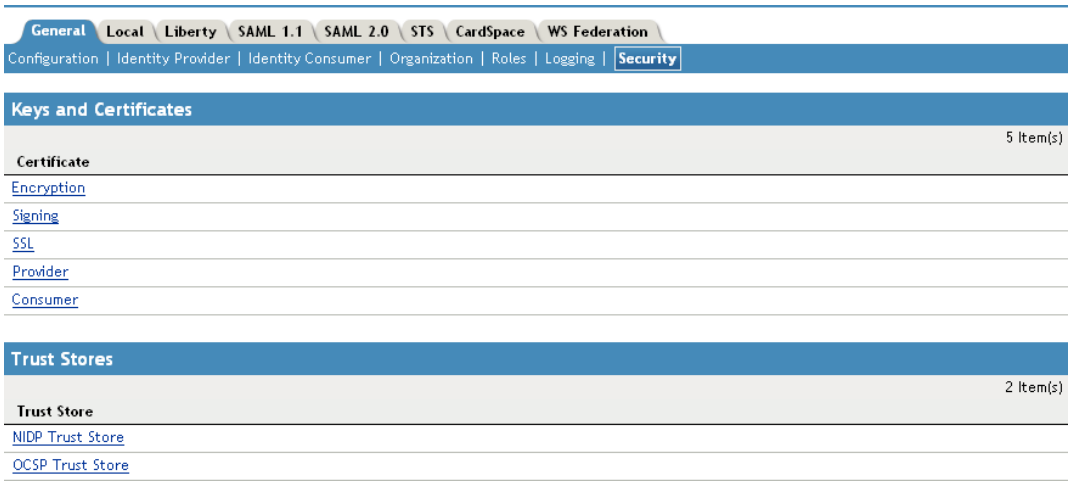
To view your current configuration:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.
2. Click the name of a profile.
3. If the *Have Discovery Encrypt This Service's Resource IDs* option is selected, the encryption key pair is used to encrypt the resource IDs.

1.3.3 Managing the Keys, Certificates, and Trust Stores

You can view the private keys, CA certificates, and certificate containers associated with the Identity Server configuration. Primarily, you use the Security page to add and replace CA certificates as necessary and to perform certificate management tasks, such as adding trusted root certificates to a trust store.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Security*.



2 To view or manage keys and certificates:

2a Click any of the following links:

Encryption: Displays the NIDP encryption certificate keystore. The encryption certificate is used to encrypt specific fields or data in the assertions. Click *Replace* to replace the encryption certificate.

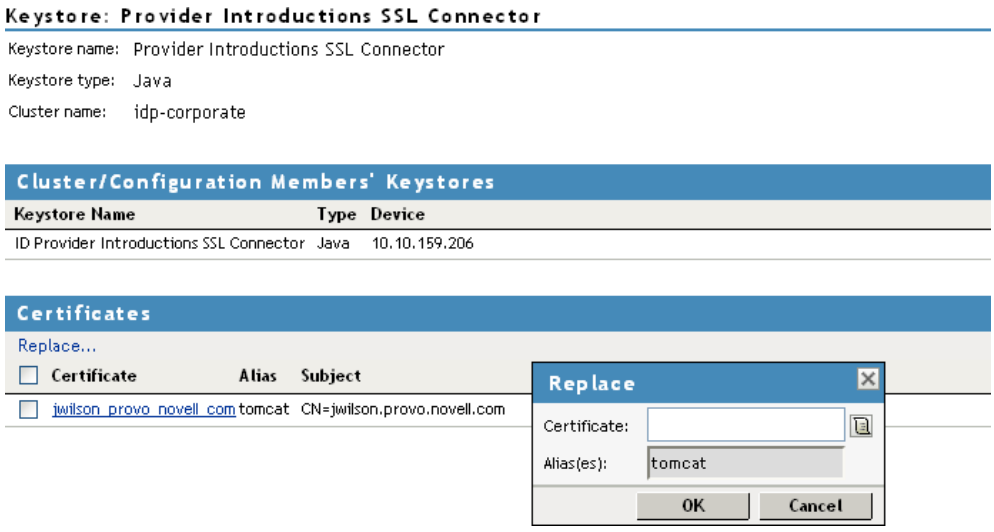
Signing: Displays the NIDP signing certificate keystore. The signing certificate is used to sign the assertion or specific parts of the assertion. Click *Replace* to replace the signing certificate.

SSL: (Required) Displays the SSL connector keystore. Click this link to access the keystore and replace the connector certificate.

Provider: Displays the ID Provider Introductions SSL Connector keystore. Click this link to access the keystore and replace the provider certificate used by the Identity Server when it is acting as an identity provider.

Consumer: Displays the ID Consumer Introductions SSL Connector keystore. Click this link to access the keystore and replace the consumer certificate used by the Identity Server when it is acting as an identity consumer (service provider).

For example, when you click the Provider keystore, the following page appears:



2b To replace a certificate, click *Replace*, browse to locate the certificate, then click *OK*.

2c If prompted to restart Tomcat, click *OK*. Otherwise, update the Identity Server.

3 To manage trust stores associated with the Identity Server:

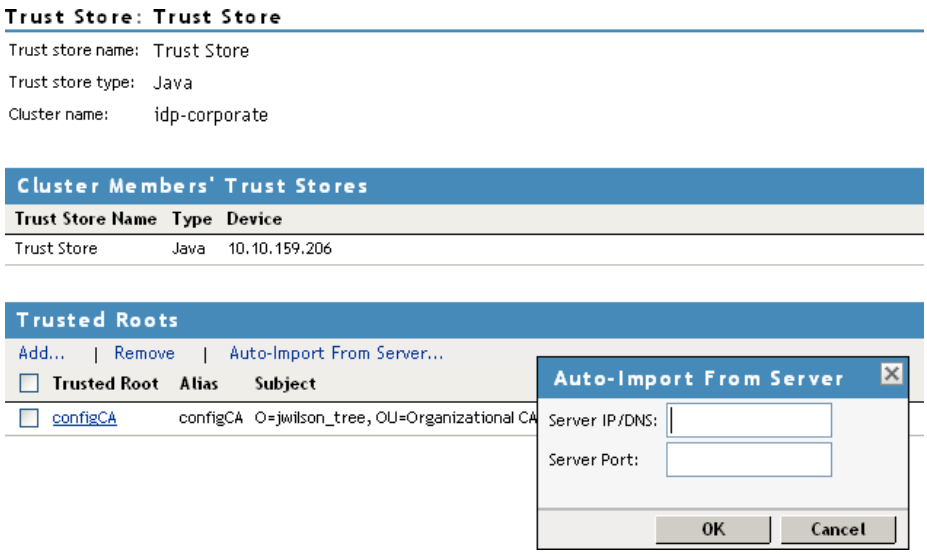
3a Click either of the following links on the Security page:

NIDP Trust Store: This Identity Server trust store contains the trusted root certificates of all the providers that it trusts. Liberty and SAML 2.0 protocol messages that are exchanged between identity and service providers often need to be digitally signed. A provider uses the signing certificate included with the metadata of a trusted provider to validate signed messages from the trusted provider. The trusted root of the CA that created the signing certificate for the service provider needs to be in this trust store.

To use SSL for protocol messages to be exchanged between providers, each provider must trust the SSL certificate authority (CA) of the other provider. You must import the root certificate chain for the other provider. Failure to do so causes numerous system errors.

OCSP Trust Store: The Identity Server uses this trust store for OCSP certificates. Online Certificate Status Protocol is a method used for checking the revocation status of a certificate. To use this feature, you must set up an OCSP server. The Identity Server sends an OCSP request to the OCSP server to determine if a certain certificate has been revoked. The OCSP server replies with the revocation status. If this revocation checking protocol is used, the Identity Server does not cache or store the information in the reply, but sends a request every time it needs to check the revocation status of a certificate. The OCSP reply is signed by the OCSP server. To verify that it was signed by the correct OCSP server, the OCSP server certificate needs to be added to this trust store. The OCSP server certificate itself is added to the trust store, not the CA certificate.

For example, if you click the NIDP Trust Store, the following page appears:



3b Select one of the following actions:

- To add a trusted root that you have already imported, click *Add*, click the *Select Trusted Roots* icon, select the trusted root, then click *OK* twice.

- To import the trusted root from the server, click *Auto-Import From Server*, specify the server's IP address or DNS name and port, then click *OK*. The auto-import displays the certificate chain, which you can select for import.
- To remove a trusted root, select the trusted root, then click *Remove*.

3c Click *Close*.

3d Update the Identity Server.

For more information about enabling security for a basic Access Manager configuration, see "Enabling SSL Communication" in the *Novell Access Manager 3.1 SP2 Setup Guide*.

For additional information about managing certificates, see "Security and Certificate Management" in the *Novell Access Manager 3.1 SP2 Administration Console Guide*.

1.4 Security Considerations

By default, all Access Manager components (Identity Server, Access Gateway, SSL VPN, and J2EE Agents) trust the certificates signed by the local CA. We recommend that you configure the Identity Server to use an SSL certificate signed externally, and that you configure the trusted store of the Embedded Service Provider for each component to trust this new CA. See "Assigning Certificates to Access Manager Devices" in the *Novell Access Manager 3.1 SP2 Administration Console Guide*.

Be aware of the following options that can increase security:

1.4.1 Federation Options

When you set up federation between an identity provider and a service provider, you can select either to exchange assertions with a post method or to exchange artifacts.

- An assertion in a post method might contain the user's password or other sensitive data, which can make it less secure than an artifact when the assertion is sent to the browser. It is possible for a virus on the browser machine to access the memory where the browser decrypts the assertion.
- An artifact is a randomly generated ID, it contains no sensitive data, and only the intended receiver can use it to retrieve assertion data.

If both providers support artifacts, you should select this method because it is more secure. For more details, see the *Response protocol binding* option in Section 7.8, "Configuring an Authentication Request for an Identity Provider," on page 211.

1.4.2 Authentication Contracts

By default, the Administration Console allows you to select from the following contracts and options when specifying whether a resource requires an authentication contract:

- **None:** Allows public access to the resource and does not require authentication contract.
- **Name/Password - Basic:** Requires that the user enter a name and password that matches an entry in an LDAP user store. The credentials do not need to be sent over a secure port. This uses the unprotected BasicClass, which is not recommended for a production environment.
- **Name/Password - Form:** Requires that the user enter a name and password that matches an entry in an LDAP user store. The credentials do not need to be sent over a secure port, although they can be if the user is configured for HTTPS. This contract uses the unprotected PasswordClass, which is not recommended for a production environment.
- **Secure Name/Password - Basic:** Requires that the user enter the name and password from a secure (SSL) connection. This uses the ProtectedBasicClass, which is recommended for a production environment. If your Web servers are using basic authentication, this contract provides the credentials for this type of authentication.
- **Secure Name/Password - Form:** Requires that the user enter the name and password from a secure (SSL) connection. This uses the ProtectedPasswordClass, which is recommended for a production environment.
- **Any Contract:** Allows the user to use any contract defined for the Identity Server configuration.

If you have set up the Access Manager to require SSL connections among all of its components, you should delete the Name/Password - Form and the Name/Password - Basic contracts. This removes them from the list of available contracts when configuring protected resources and prevents them from being assigned as the contract for a protected resource. If these contracts are assigned, the user's password can be sent across the wire in clear text format. At some future date, if your system needs this type of contract, you can re-create it from the method. To delete these contracts, go to the Administration Console and click *Identity Servers* > *Servers* > *Edit* > *Local* > *Contracts*.

1.4.3 Forcing 128-Bit Encryption

You can force all client communication with the Identity Server to use 128-bit encryption by modifying the `server.xml` file used by Tomcat. If the browser is unable to supported the encryption level specified in this file, the user is not allowed to authenticate.

1. At a command prompt, change to the Tomcat configuration directory:

   Linux: `/var/opt/novell/tomcat5/conf`

   Windows Server 2003: `\Program Files\Novell\Tomcat\conf`

   Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\conf`

2. To the `server.xml` file, add the cipher suites you want to support. For 128-bit encryption, add the following line:

   ```
   ciphers="TLS_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
   TLS_DHE_DSS_WITH_AES_128_CBC_SHA"
   ```

   This is a comma-separated list of the JSSE names for the TLS cipher suites.

IMPORTANT: If you enter a cipher name incorrectly, Tomcat reverts to the default values, which allow the weak ciphers to be used.

If you want to allow the SSL cipher suites, the following JSSE names can be added to the list:

```
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
```

For a complete list of supported cipher suites and their requirements, see The SunJSSE Provider (http://java.sun.com/javase/6/docs/technotes/guides/security/SunProviders.html#SunJSSEProvider).

3 To activate the cipher list, restart Tomcat.

Linux: Enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

Windows: Enter the following commands:

```
net stop Tomcat5
net start Tomcat5
```

4 (Conditional) If you have multiple Identity Servers in your cluster configuration, repeat these steps on each Identity Server.

1.4.4 Securing the Identity Server Cookie

An attacker can spoof a non-secure browser into sending a JSESSION cookie that contains a valid user session. To stop this from happening, you need to first configure the Identity Server to use SSL. For configuration information, see "Configuring Secure Communication on the Identity Server" in the *Novell Access Manager 3.1 SP2 Setup Guide*.

After you have configured the Identity Server to use SSL, you need to configure Tomcat to secure the cookie.

1 On the Identity Server, log in as the administrator.

2 Change to the Tomcat configuration directory:

Linux: `/var/opt/novell/tomcat5/conf`

Windows Server 2003: `\Program Files\Novell\Tomcat\conf`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\conf`

3 Create a `context.xml` file with the following content:

```
<Context useHttpOnly="true">
</Context>
```

4 Save the file, then restart Tomcat:

Linux: Enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

Windows: Enter the following commands:

```
net stop Tomcat5
net start Tomcat5
```

1.4.5 Configuring the Encryption Method for the SAML Assertion

By default, AES128 (Advanced Standard Encryption, 128-bit) is used to encrypt SAML assertions. If you require a different encryption method, such as TDES (Triple Data Encryption Algorithm) or AES256 (Advanced Standard Encryption, 256-bit), you can modify the Tomcat `web.xml` file and specify your required method.

1. Open the `web.xml` file.

   Linux: `/var/opt/novell/tomcat5/webapps/nidp/WEB-INF/`

   Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\WEB-INF/`

   Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF/`

2. Add the following lines to the file:

   ```
   <context-param>
           <param-name>EncryptionMethod</param-name>
           <param-value>TDES</param-value>
   </context-param>
   ```

   You can set the `<param-value>` element to TDES, AES128, or AES256. Because AES128 is the default, specifying this value in the `web.xml` file does not change any behavior.

3. Save the file and copy it to each Identity Server in the cluster.

4. Restart Tomcat on each Identity Server in the cluster.

   Linux: Enter the following command:

   ```
   /etc/init.d/novell-tomcat5 restart
   ```

   Windows: Enter the following commands:

   ```
   net stop Tomcat5
   ```

   ```
   net start Tomcat5
   ```

1.4.6 Configuring SAML 2.0 to Sign Messages

In conformance with the SAML 2.0 specification, the Identity Server does not require the signing post messages. However, if you want this extra layer of security, you can configure the Identity Server to sign SAML 2.0 post messages. This is a global option, and when enabled, all SAML 2.0 service providers sign post messages.

To enable the signing of post messages:

1. Open the `web.xml` file.

   Linux: `/var/opt/novell/tomcat5/webapps/nidp/WEB-INF/`

   Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\WEB-INF/`

   Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF/`

2. Add the following lines to the file:

```
<context-param>
   <param-name>SignPost</param-name>
   <param-value>true</param-value>
</context-param>
```

3 Save the file and copy it to each Identity Server in the cluster.

4 Restart Tomcat on each Identity Server in the cluster.

Linux: Enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

Windows: Enter the following commands:

```
net stop Tomcat5
```

```
net start Tomcat5
```

1.4.7 Blocking Access to Identity Server Pages

The Identity Server has a couple of pages that authenticated users can access and which contain information about the user and the Identity Server that some security models deem sensitive. If you want to block user access to these pages, see the following sections:

- Section 3.6.3, “Blocking Access to the User Portal Page,” on page 136
- Section 3.6.4, “Blocking Access to the WSDL Services Page,” on page 137

1.5 Translating the Identity Server Configuration Port

If your Identity Server must communicate through a firewall, you must either set up a hole in your firewall for TCP ports 8080 or 8443 (default ports used respectively for non secure and secure communication with Identity Server), or configure the Identity Server service to use TCP port 80 or 443.

- Section 1.5.1, “Changing the Port on a Windows Identity Server,” on page 36
- Section 1.5.2, “Changing the Port on a Linux Identity Server,” on page 37

1.5.1 Changing the Port on a Windows Identity Server

On a Windows Identity Server, you need to set the port in the Base URL and save the changes. You then need to modify the Tomcat `server.xml` file located in the Tomcat configuration directory:

1 In the Administration Console, click *Devices* > *Identity Server* > *Edit*, and configure the base URL with HTTPS as the protocol, and the TCP port as 443.

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

3 In a terminal window, open the `server.xml` file.

Windows Server 2003: `\Program Files\Novell\Tomcat\conf`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\conf`

4 Change the ports from 8080 and 8443 to 80 and 443.

5 Restart the Tomcat service.

```
net stop Tomcat5
net start Tomcat5
```

1.5.2 Changing the Port on a Linux Identity Server

On a Linux Identity Server, the Identity Server service (hosted on Tomcat) runs as a non-privileged user on Linux and cannot therefore bind to ports below 1024. In order to allow requests to port 80/443 while Tomcat is listening on 8080/8443, the preferred approach is to use iptables to perform a port translation. Port translation allows the base URL of the Identity Server to be configured for port 443 and to listen on this port, and the iptables translates it to port 8443 when communicating with Tomcat.

- If you have disabled the SUSE Linux Enterprise Server (SLES) firewall and do not have any other Access Manager components installed on the Identity Server, you can use a simple iptables script to translate the ports. See "A Simple Redirect Script" on page 37.
- If you have configured the SLES firewall or have installed other Access Manager components on the Identity Server, you use a custom rule script that allows for multiple port translations. See "Configuring iptables for Multiple Components" on page 39.

These sections describe two solutions out of many possibilities. For more information about iptables, see the following:

- "Iptable Tutorial 1.2.2" (http://iptables-tutorial.frozentux.net/iptables-tutorial.html)
- "NAM Filters for iptables Commands" (http://www.novell.com/communities/node/4029/nam-filters-iptables-commands)

A Simple Redirect Script

This simple solution works only if you are not using iptables to translate ports of other applications or Access Manager components. For a solution that works with multiple components, see "Configuring iptables for Multiple Components" on page 39.

1. In the Administration Console, click *Devices* > *Identity Server* > *Edit*, and configure the base URL with HTTPS as the protocol, and the TCP Port as 443.
2. Click *OK*, then update the Identity Server.
3. At a terminal window, log in as the `root` user.
4. Create a file to hold the iptables rule and place it in the `/etc/init.d` directory.

   For example, `/etc/init.d/AM_IDP_Redirect`. Ensure it has execute rights. You can use CHMOD as appropriate.

   An example of a redirect startup file for this purpose might be:

```
#!/bin/sh
# Copyright (c) 2010 Novell, Inc.
# All rights reserved.
#
#! /bin/sh
#! /etc/init.d/idp_8443_redirect
# ### BEGIN INIT INFO
# Provides: idp_8443_redirect
# Required-Start:
# Required-Stop:
# Default-Start: 2 3 5
```

```
# Default-Stop: 0 1 6
# Description: Redirect 8443 to 443 for Novell IDP
### END INIT INFO #

# Environment-specific variables.
IPT_BIN=/usr/sbin/iptables
INTF=eth0
ADDR=10.10.0.1

. /etc/rc.status


# First reset status of this service
rc_reset

case "$1" in
    start)
        echo -n "Starting IP Port redirection"
        $IPT_BIN -t nat --flush
        $IPT_BIN -t nat -A PREROUTING -i $INTF -p tcp --dport 80 -j DNAT -
-to ${ADDR}:8080
        $IPT_BIN -t nat -A PREROUTING -i $INTF -p tcp --dport 443 -j DNAT
--to ${ADDR}:8443
       $IPT_BIN -t nat -A OUTPUT -p tcp -d $ADDR --dport 443 -j DNAT --to
${ADDR}:8443
        $IPT_BIN -t nat -A OUTPUT -p tcp -d $ADDR --dport 80 -j DNAT --to
${ADDR}:8080
        rc_status -v
        ;;
    stop)
        echo -n "Flushing all IP Port redirection rules"
        $IPT_BIN -t nat --flush
        rc_status -v
        ;;
    restart)
        $0 stop
        $0 start
        rc_status
        ;;
    *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
        ;;
esac
rc_exit
```

For more information about init scripts for SUSE Linux Enterprise Server 10, see "Section 20.2.2 Init Scripts" (http://www.novell.com/documentation/sles10/book_sle_reference/data/sec_boot_init.html) in the *SUSE Linux Enterprise Server 10 Installation and Administration Guide* (http://www.novell.com/documentation/sles10/index.html).

For more information about init scripts for SUSE Linux Enterprise Server 11, see "Section 7.2.2 Init Scripts" (http://www.novell.com/documentation/sles11/book_sle_admin/data/sec_boot_init.html) in the *SLES 11 Administration Guide* (http://www.novell.com/documentation/sles11/book_sle_admin/data/book_sle_admin_pre.html).

5 Modify the environment-specific variables found in the following lines:

```
# Environment-specific variables.
IPT_BIN=/usr/sbin/iptables
INTF=eth0
ADDR=10.10.0.1
```

6 To ensure that the iptables rule is active after rebooting, start YaST, click *System*, > *System Services (Runlevel)*, select *Expert Mode*, select the file you created, enable runlevels boot, 3 and 5 for the file, then start the service.

7 To verify that your script is running, enter the following command:

```
ls /etc/init.d/rc3.d | grep -i AM_IDP_Redirect
```

8 Reboot the Identity Server machine.

9 After rebooting, verify that port 443 is being routed to the Identity Server by entering the following command:

```
iptables -t nat -nvL
```

You should see an entry similar to the following:

```
pkts bytes target     prot opt in     out     source
destination
17   748    DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/
0           tcp dpt:443 to:10.10.0.1:8443
```

This entry states that eth0 is routing TCP port 443 to IP address 10.10.0.1.

10 (Conditional) If your Identity Server cluster configuration contains more than one Identity Server, repeat these steps on each server in the cluster.

Configuring iptables for Multiple Components

If you need to use iptables for multiple components (the host machine, the Identity Server, or the SSL VPN server), you need to centralize the commands into one manageable location. The following sections explain how to use the SuSEFirewall2 option in YaST to centralize the commands.

The Identity Server and the SSL VPN server use different routing methods, so their commands are different. The Identity Server requires pre-routing commands, and the SSL VPN server uses post-routing commands.

- “Adding the Identity Server Commands” on page 39
- “Adding the SSL VPN Commands” on page 40

Adding the Identity Server Commands

1 In the Administration Console, click *Devices* > *Identity Server* > *Edit*, and configure the base URL with HTTPS as the protocol, and the TCP port as 443.

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

3 On the Identity Server, edit the `/etc/sysconfig/SuSEfirewall2` file.

3a Change the FW_CUSTOMRULES="" line to the following:

```
FW_CUSTOMRULES="/etc/sysconfig/scripts/SuSEfirewall2-custom"
```

3b Save the changes and exit.

4 Open the `/etc/sysconfig/scripts/SuSEfirewall2-custom` file in an editor.

This is the custom rules file you specified in Step 3.

5 Add the following lines under the `fw_custom_before_port_handling()` section:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j DNAT --to
10.10.0.1:8443
iptables -t nat -A OUTPUT -p tcp -o eth0 --dport 443 -j DNAT --to
10.10.0.1:8443
true
```

The first command rewrites all incoming requests with a destination TCP port of 443 to TCP port 8443 on the 10.10.0.1 IP address for eth0. Modify the IP address to match the IP address of your Identity Server.

The second command rewrites the health checks.

6 Select one of the following:

- If you need to add commands for the SSL VPN server, continue with "Adding the SSL VPN Commands" on page 40.
- If you don't need to add any other commands, save the file, then continue with Step 7.

7 At the system console, restart the firewall by executing the following command:

```
/etc/init.d/SuSEfirewall2_setup restart
```

8 After rebooting, verify that port 443 is being routed to the Identity Server by entering the following command:

```
iptables -t nat -nvL
```

You should see an entry similar to the following:

```
pkts bytes target     prot opt in     out     source
destination
17   748 DNAT     tcp  --  eth0   *      0.0.0.0/0          0.0.0.0/0
tcp dpt:443 to:10.10.0.1:8443
```

This entry states that eth0 is routing TCP port 443 to IP address 10.10.0.1:8443.

9 (Conditional) If your Identity Server cluster configuration contains more than one Identity Server, repeat these steps on each server in the cluster.

Adding the SSL VPN Commands

These steps assume that you have completed at least Step 3 in "Adding the Identity Server Commands" on page 39.

1 Add the following lines to the `fw_custom_before_masq` section of the `/etc/sysconfig/scripts/SuSEfirewall2-custom` file.

```
iptables -t nat -A POSTROUTING -s 10.8.0.0/16 -j SNAT --to 10.1.1.1
```

The 10.8.0.0/16 address is configured as a tunnel subnet, and the 10.1.1.1 address is your private interface.

2 Add the following lines to the `fw_custom_before_denyall` section.

```
iptables -A $chain -j ACCEPT -s 10.8.0.0/22
iptables -A $chain -j ACCEPT -d 10.8.0.0/22
```

The file should look similar to the following:

```
fw_custom_before_masq() {
   iptables -t nat -A POSTROUTING -s 10.8.0.0/16 -j SNAT --to 10.1.1.1
true
}

fw_custom_before_denyall() {
  for chain in input_ext input_dmz input_int forward_int forward_ext
forward_dmz; do
  iptables -A $chain -j ACCEPT -s 10.8.0.0/22
  iptables -A $chain -j ACCEPT -d 10.8.0.0/22
  done

  true
}
```

3. Save the file.
4. Restart the firewall by executing the following command:

   ```
   /etc/init.d/SuSEfirewall2_setup restart
   ```

5. Verify that the post SSL VPN routing iptables filters have been registered correctly by issuing the following command:

   ```
   iptables -t nat -nvL
   ```

   You should see information similar to the following if the filters have been registered correctly:

   ```
   Chain POSTROUTING (policy ACCEPT 20987 packets, 1266K bytes)
   pkts bytes target prot opt in  out  source       destination
    0     0    SNAT   all  --  *   *    10.8.0.0/16   0.0.0.0/0
   to:10.1.1.1
   ```

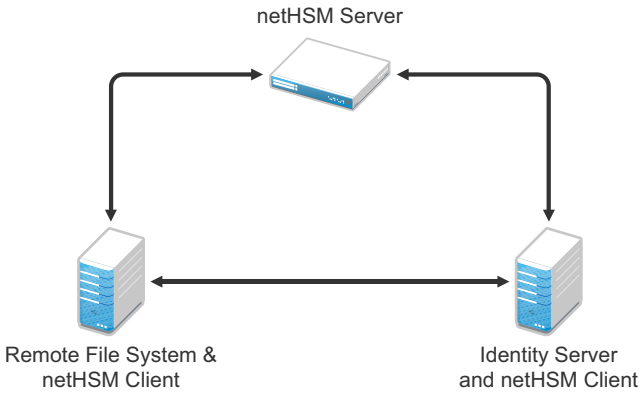
1.6 Using netHSM for the Signing Key Pair

netHSM is a Hardware Security Module (HSM) from nCipher. The module is attached to the network and provides cryptographic resources for multiple servers. Keys stored in a netHSM keystore are secure because the key material can never be exposed outside of the module.

Access Manager has not been tested with any other HSM products; it has only been tested with the netHSM module from nCipher.

Figure 1-2 illustrates a simple netHSM configuration with an Identity Server as a netHSM client.

Figure 1-2 *A Simple netHSM Configuration*



Access Manager allows you to use netHSM to store and manage the signing key pair of the Identity Server. You must use the Administration Console to store and manage the other Access Manager certificates. Access Manager uses the Java Security provider of the netHSM server to interact with the netHSM server.

This section describes the following about the netHSM implementation:

- Section 1.6.1, "Understanding How Access Manager Uses Signing and Interacts with the netHSM Server," on page 42
- Section 1.6.2, "Configuring the Identity Server for netHSM," on page 44

1.6.1 Understanding How Access Manager Uses Signing and Interacts with the netHSM Server

The netHSM server provides a signing certificate that is used instead of the one provided by Access Manager. Requests, responses, assertions, or payloads can be signed when there are interactions during single sign-on or during attribute queries between service providers and identity providers using any of the SAML1.1, SAML2, Liberty ID-FF, Liberty ID-WSF, or ID-SIS protocols.

- "Access Manager Services That Use the Signing Certificate" on page 42
- "Understanding the Interaction of the netHSM Server with Access Manager" on page 43

Access Manager Services That Use the Signing Certificate

The following services can be configured to use signing:

- "Protocols" on page 42
- "SOAP Back Channel" on page 42
- "Profiles" on page 43

Protocols

The protocols can be configured to sign authentication requests.

To view your current configuration:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit*.
2. In the *Identity Provider* section, view the setting for the *Require Signed Authentication Requests* option. If it is selected, all authentication requests from identity providers are signed.
3. In the *Identity Consumer* section, view the settings for the *Require Signed Assertion*s and S*ign Authentication Requests* options. If these options are selected, assertions and authentication requests are signed.

SOAP Back Channel

The SOAP back channel is the channel that the protocols use to communicate directly with a provider. The SOAP back channel is used for artifact resolutions and attribute queries for the Identity Web Services Framework.

To view your current configuration for the SOAP back channel:

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

2 Select the protocol (Liberty, SAML 1.1, or SAML 2.0), then click the name of an identity provider or service provider.

3 Click *Access*.

4 View the *Security* section. If the *Message Signing* option is selected, signing is enabled for the SOAP back channel.

Profiles

Any of the Web Service Provider profiles can be enabled for signing by configuring them to use X.509 for their security mechanism.

To view your current configuration:

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.

2 Click the name of a profile, then click *Descriptions*.

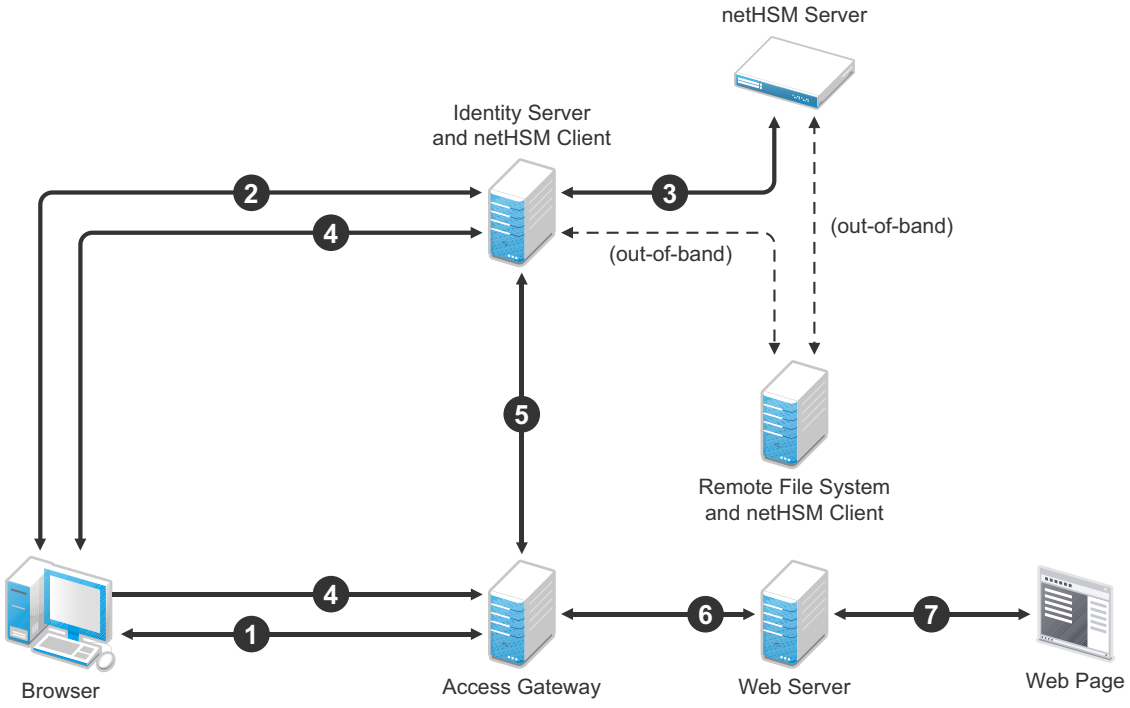
3 Click the *Description Name*.

4 If either *Peer entity = None, Message=X509* or *Peer entity = MutualTLS, Message=X509* has been selected as the security mechanism, signing has been enabled for the profile.

Understanding the Interaction of the netHSM Server with Access Manager

Figure 1-3 outlines one of the basic flows that might occur during single sign-on to the Identity Server when authentication requests have been configured for signing.

Figure 1-3 *Basic Flow for an Authentication Request Using netHSM*



1. The user requests the Access Gateway to provide access to a protected resource.

2. The Access Gateway redirects the user to the Identity Server, which prompts the user for a username and password.
3. The Identity Server authenticates the user. If signing is enabled, the payload is signed by the netHSM server through the Java JSSE security provider.
4. The Identity Server returns the authentication artifact to the Access Gateway.
5. The Embedded Service Provider of the Access Gateway retrieves the user's credentials from the Identity Server.
6. The Access Gateway verifies that the credentials allow the user access to the resource, then sends the request to the Web server.
7. The Web server returns the requested Web page.

1.6.2 Configuring the Identity Server for netHSM

Prerequisites for Using netHSM

- ❑ An installed and configured netHSM server.
- ❑ An installed and configured remote file system with the netHSM client.
- ❑ An installed Identity Server, assigned to a cluster configuration.

  For instructions on a basic setup that assigns the Identity Server to a cluster configuration, see "Creating a Basic Identity Server Configuration" in the *Novell Access Manager 3.1 SP2 Setup Guide*.

The following instructions describe one way to integrate the Identity Server with a netHSM server. Other ways are possible.

Configuring the Identity Server to Be a netHSM Client

The following instructions are based on nCipher hardware, but you should be able to adapt them for your hardware. The instructions explain how to configure the Identity Server so that it can communicate with both the nCipher server and the remote file system server, how to create a signing key pair and its keystore, how to copy these them to the Identity Server, and how to synchronize the changes with the remote file system server.

1. At the Identity Server, log in as the root or administrator user and install the netHSM client software.

   The nCipher software installs files in the `/opt/nfast` directory on Linux and in the `C:\nfast` directory on Windows. It creates an nfast user and group. Check your netHSM documentation for the specific steps.
2. (Conditional) If your Identity Server cluster configuration contains more than one Identity Server, install the netHSM client software on the other Identity Servers in the cluster.

3 At the netHSM server, configure the server to allow the Identity Server to be a client.

Check your netHSM documentation for the specific steps.

4 (Conditional) If your Identity Server cluster configuration contains more than one Identity Server, configure the netHSM server to allow the other Identity Servers in the cluster to be a client.

5 At the Identity Server, enroll the client to use the server:

5a To get the ESN and hash numbers for the enroll command, enter the following command:

Linux: `/opt/nfast/bin/anonkneti <IP_address>`

Windows: `C:\nfast\bin>anonkneti <IP_address>`

Replace <*IP_address*> with the IP address of the netHSM server.

5b To enroll the client, enter the following command:

Linux: `/opt/nfast/bin/nethsmenroll -p <IP_address> <ESN> <hash>`

Windows: `C:\nfast\bin>nethsmenroll -p <IP_address> <ESN> <hash>`

Replace <*IP_address*> with the IP address of the netHSM server. Replace <*ESN*> and <*hash*> with the values copied from the `anonkneti` command.

6 (Conditional) If the Identity Server and the Administration Console are installed on the same machine, modify the 9000 and 9001 TCP ports:

6a In a text editor, open the `sc.conf` file located in the following directory:

Linux: `/opt/novell/devman/share/conf`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\roma\WEB-INF\conf`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\roma\WEB-INF\conf`

6b Change the ports from 9000 and 9001 to another value, such as 9010 and 9011.

The lines should look similar to the following:

```
<stringParam name="ExecutorPort" value="9010" />
<stringParam name="SchedulerPort" value="9011" />
```

6c Save the changes.

6d Restart Tomcat:

Linux: Enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

Windows: Enter the following commands:

```
net stop Tomcat5
net start Tomcat5
```

6e (Conditional) If other Identity Servers in the cluster contain an Administration Console, repeat Step 6.

7 At the Identity Server, enable the netHSM client so that it uses TCP:

7a Enter the following command:

Linux: `/opt/nfast/bin/config-serverstartup -sp`

Windows: `C:\nfast\bin>config-serverstartup -sp`

7b To restart the nfast client:

Linux: Enter the following command:

```
/opt/nfast/sbin/init.d-nfast restart
```

Windows: Enter the following commands:

```
C:\nfast\bin>net stop "nfast server"
```

```
C:\nfast\bin>net start "nfast server"
```

8 Configure communication to the remote file system server. In this sample configuration, the remote file system is installed on a Windows machine.

8a At the remote file system server, enable communication with the Identity Server. For a Windows machine, enter the following command:

```
C:\nfast\bin\rfs-setup.exe --gang-client --write-noauth <address>
```

Replace <*address*> with the IP address of the Identity Server.

8b At the Identity Server, enable communication with the remote file system server. For nCipher, enter the following command:

Linux: `/opt/nfast/bin/rfs-sync --setup --no-authenticate <address>`

Windows: `C:\nfast\bin>rfs-sync --setup --no-authenticate <address>`

Replace <*address*> with the IP address of the remote file system server.

8c At the Identity Server, initialize synchronization with the remote file system server.

Linux: Enter the following commands:

```
/opt/nfast/bin/rfs-sync --update
```

```
/opt/nfast/bin/rfs-sync --commit
```

Windows: Enter the following commands:

```
C:\nfast\bin>rfs-sync --update
```

```
C:\nfast\bin>rfs-sync --commit
```

The first command reads updates from the remote file system server and downloads files to the `/opt/nfast/kmdata/local` directory on Linux and the `C:\nfast\kmdata\local` directory on Windows. The second command writes local changes to the remote file system server.

9 Continue with "Creating the nCipher Signing Key Pair" on page 46.

Creating the nCipher Signing Key Pair

IMPORTANT: Because of Access Manager configuration conflicts, you need to use a netHSM client other than the Identity Server. The remote file system server is a netHSM client, or if you have configured another device as a client, you can use that device.

The following commands are specific to nCipher; it does not come with a tool to generate a key pair and CSR. nCipher also uses a unique keystore of type `nCipher.sworld`.

nCipher supports both a Windows and a Linux netHSM client.

- If you have a Windows netHSM client, the command is located in the following directory:

  ```
  c:\Program Files\Java\jdk1.5.0_14\jre\bin\java
  ```

- If you have Linux netHSM client, the command is located in the following directory:

  ```
  /opt/novell/java/bin/java
  ```

To create a new key pair for nCipher:

1. On a netHSM client, add the nCipher provider to the provider list of the `java.security` file:

   1a. In a text editor, open the `C:\Program Files\Java\jdk1.5.0_14\jre\lib\security\java.security` file.

   1b. Add the following lines to the top of the list of providers:

   ```
   security.provider.1=com.ncipher.fixup.provider.nCipherRSAPrivateEncry
   pt
   security.provider.2=com.ncipher.provider.km.nCipherKM
   ```

   The provider section should look similar to the following:

   ```
   #
   # List of providers and their preference orders (see above):
   #
   security.provider.1=com.ncipher.fixup.provider.nCipherRSAPrivateEncry
   pt
   security.provider.2=com.ncipher.provider.km.nCipherKM
   security.provider.3=sun.security.provider.Sun
   security.provider.4=sun.security.rsa.SunRsaSign
   security.provider.5=com.sun.net.ssl.internal.ssl.Provider
   security.provider.6=com.sun.crypto.provider.SunJCE
   security.provider.7=sun.security.jgss.SunProvider
   security.provider.8=com.sun.security.sasl.Provider
   ```

   1c. Save your changes.

2. Add the nfast libraries to the CLASSPATH for Java:

   For a Windows client, add the following paths:

   ```
   c:\nfast\java\classes\keysafe.jar;c:\nfast\java\classes\nfjava.jar
   ;c:\nfast\java\classes\kmjava.jar;c:\nfast\java\classes\kmcsp.jar;
   c:\nfast\java\classes\jutils.jar;c:\nfast\java\classes\jcetools.
   jar;c:\nfast\java\classes\spp.jar;c:\nfast\java\classes\rsaprivenc
   .jar;
   ```

   For a Linux client, add the following paths and export them:

   ```
   /opt/nfast/java/classes/nfjava.jar:/opt/nfast/java/classes/
   kmjava.jar:/opt/nfast/java/classes/kmcsp.jar:/opt/nfast/java/
   classes/spp.jar:/opt/nfast/java/classes/rsaprivenc.jar:/opt/nfast/
   java/classes/jutils.jar:/opt/nfast/java/classes/jcetools.jar:/opt/
   nfast/java/classes/keysafe.jar
   ```

3. Create a directory for the keystore and change to that directory.

4. On a Windows client, enter the following command to create a new key in a keystore:

   ```
   "c:\Program Files\Java\jdk1.5.0_14\jre\bin\java" -Dprotect=module
   -DignorePassphrase=true sun.security.tools.KeyTool -genkey -v
   -alias od93 -keyalg RSA -keystore AMstore.jks -storetype
   nCipher.sworld -provider com.ncipher.provider.km.nCipherKM
   ```

   Enter your values for the following parameters:

Parameter	Description
`-Dprotect=module`	Required if you want the keystore to be module protected.
`-DignorePassphrase=true`	Required if you want the keystore to be module protected.
`sun.security.tools.KeyTool`	The name of the keytool command
`-alias`	A name that helps you identify the key. In this sample configuration, the name is `od93`.
`-keyalg`	The security algorithm.
`-keystore`	A name for the keystore. In this sample configuration, the name is `AMstore.jks`.
`-storetype`	The type of keystore. For nCipher, this must be set to `nCipher.sworld`.
`-provider`	The name of the providerClass and providerName. This is the provider that you added to the `java.security` file in Step 1.

The tool prompts you for a password for the keypass and the storepass. They must be the same password if you are going to use card set protection rather than module protection.

The tool also prompts you for the certificate subject name (first name, last name, organization, organizational unit, locality, state or providence, and country).

5 To generate a certificate request from a key in the keystore, enter the following command:

```
"c:\Program Files\Java\jdk1.5.0_14\jre\bin\java" -Dprotect=module
-DignorePassphrase=true sun.security.tools.KeyTool -certreq -alias
od93 -file cert.csr -keypass mypwd -keystore AMstore.jks -storepass
mypwd -storetype nCipher.sworld -provider
com.ncipher.provider.km.nCipherKM
```

Enter your values for the following parameters:

Parameter	Description
`-Dprotect=module`	Required if you want the keystore to be module protected.
`-DignorePassphrase=true`	Required if you want the keystore to be module protected.
`sun.security.tools.KeyTool`	The name of the keytool command
`-certreq`	The parameter that makes this a certificate request.
`-alias`	A name that helps you identify the certificate request. In this sample configuration, the name is `od93`.
`-file`	The name to be given to the certificate signing request file. In this sample configuration, the name is `cert.csr`.

Parameter	Description
`-keypass`	The password for the key. In this sample configuration, the password is `mypwd`.
`-keystore`	A name for the keystore. In this sample configuration, the name is `AMstore.jks`.
`-storepass`	The password for the keystore. In this sample configuration, the password is `mypwd`.
`-storetype`	The type of keystore. For nCipher, this must be set to `nCipher.sworld`.
`-provider`	The name of the providerClass and providerName.

6 Take the CSR created in Step 5 to a certificate authority. The CA needs to send you a DER-encoded public certificate. The CA also needs to send you the public certificate that it used to create the certificate and the public certificates for any CAs in the chain.

7 Load the public certificate of the CA into the keystore by entering the following command:

```
"c:\Program Files\Java\jdk1.5.0_14\jre\bin\java" -Dprotect=module
-DignorePassphrase=true sun.security.tools.KeyTool -import -alias
publicca -file certca.cer -keystore Amstore.jks -storetype
nCipher.sworld -provider com.ncipher.provider.km.nCipherKM
```

Enter your values for the following parameters:

Parameter	Description
`-Dprotect=module`	Required if you want the keystore to be module protected.
`-DignorePassphrase=true`	Required if you want the keystore to be module protected.
`sun.security.tools.KeyTool`	The name of the keytool command
`-import`	The parameter that makes this an import request.
`-alias`	A name that helps you identify that this is the public certificate from the CA. In this sample configuration, the name is `publicca`.
`-file`	The name of the CA certificate file. In this sample configuration, the name is `certca.cer`.
`-keystore`	A name for the keystore. In this sample configuration, the name is `AMstore.jks`.
`-storetype`	The type of keystore. For nCipher, this must be set to `nCipher.sworld`.
`-provider`	The name of the providerClass and providerName.

The tool prompts you for the keystore password and asks whether you want to trust the certificate.

8 (Conditional) Repeat Step 7 for each CA in the chain, giving each CA a unique alias.

9 Import the signed certificated received from the CA by entering the following command:

```
"c:\Program Files\Java\jdk1.5.0_14\jre\bin\java" -Dprotect=module
-DignorePassphrase=true sun.security.tools.KeyTool -import -alias
od93 –file signcert.der -keystore AMstore.jks -storepass mypwd
-storetype nCipher.sworld -provider
com.ncipher.provider.km.nCipherKM
```

Enter your values for the following parameters:

Parameter	Description
`-Dprotect=module`	Required if you want the keystore to be module protected.
`-DignorePassphrase=true`	Required if you want the keystore to be module protected.
`sun.security.tools.KeyTool`	The name of the keytool command
`-import`	The parameter that makes this an import request.
`-alias`	A name that helps you identify that this is the signing key pair from the CA. It needs to be the same alias you specified when you created the keystore in Step 4. In this sample configuration, the name is `od93`.
`-file`	The name of the signing certificate file from the CA. In this sample configuration, the name is `signcert.der`.
`-keystore`	A name for the keystore. In this sample configuration, the name is `AMstore.jks`.
`-storepass`	The password for the keystore. In this sample configuration, the password is `mypwd`.
`-storetype`	The type of keystore. For nCipher, this must be set to `nCipher.sworld`.
`-provider`	The name of the providerClass and providerName.

10 (Optional) To verify that the certificates have been added to the keystore, enter the following command:

```
"c:\Program Files\Java\jdk1.5.0_14\jre\bin\java" -Dprotect=module
-DignorePassphrase=true sun.security.tools.KeyTool -list -v
-keystore AMstore.jks -storetype nCipher.sworld -provider
com.ncipher.provider.km.nCipherKM
```

The keystore should contain at least two certificates. The certificate that you created should now be issued by the CA you used, and the public certificate of the CA should be there as the owner and the issuer.

11 Copy the keystore to the `idp` directory on the Identity Server.

Linux: `/opt/novell/devman/jcc/certs/idp`

Windows Server 2003: `\Program Files\Novell\devman\jcc\certs\idp`

Windows Server 2008: `\Program Files (x86)\Novell\devman\jcc\certs\idp`

The keystore is found on the netHSM client in the directory specified by the `-keystore` parameter when you created the keystore. See Step 4.

12 Synchronize the Identity Server with the remote file system server.

Linux: Enter the following commands:

```
/opt/nfast/bin/rfs-sync --update
/opt/nfast/bin/rfs-sync --commit
```

Windows: Enter the following commands:

```
C:\nfast\bin>rfs-sync --update
C:\nfast\bin>rfs-sync --commit
```

13 (Conditional) If the cluster configuration contains more than one Identity Server, complete the following steps for each cluster member:

13a Copy the keystore to the cluster member. Copy it to the following directory:

Linux: `/opt/novell/devman/jcc/certs/idp`

Windows Server 2003: `\Program Files\Novell\devman\jcc\certs\idp`

Windows Server 2008: `\Program Files (x86)\Novell\devman\jcc\certs\idp`

13b Make sure the `novlwww` user has at least read rights.

13c Use the netHSM client to synchronize the cluster member with the remote file system server.

Linux: Enter the following commands:

```
/opt/nfast/bin/rfs-sync --update
/opt/nfast/bin/rfs-sync --commit
```

Windows: Enter the following commands:

```
C:\nfast\bin>rfs-sync --update
C:\nfast\bin>rfs-sync --commit
```

14 Continue with "Configuring the Identity Server to Use the netHSM Certificate" on page 51.

Configuring the Identity Server to Use the netHSM Certificate

The procedure to modify the classpath names depends upon whether you have a Linux or a Windows Identity Server:

- "Configuring a Linux Identity Server for the Certificate" on page 51
- "Configuring a Windows Identity Server for the Certificate" on page 53

Configuring a Linux Identity Server for the Certificate

1 At the Identity Server, log in as `root`.

2 Add the nfast jar files to the classpath.

Because the Identity Server runs as a Tomcat service, the following steps explain how to modify the classpath for Tomcat.

2a In an editor, open the `/opt/novell/tomcat5/bin/dtomcat5` file.

2b To the `CLASSPATH="$JAVA_HOME"/lib/tools.jar` line, add the following classes from the `/opt/nfast/java/classes` directory:

```
nfjava.jar
kmjava.jar
kmcsp.jar
spp.jar
rsaprivenc.jar
jutils.jar:
jcetools.jar
keysafe.jar
```

Your line should look similar to the following:

```
CLASSPATH="$JAVA_HOME"/lib/tools.jar:/opt/nfast/java/classes/
nfjava.jar:/opt/nfast/java/classes/kmjava.jar:/opt/nfast/java/
classes/kmcsp.jar:/opt/nfast/java/classes/spp.jar:/opt/nfast/
java/classes/rsaprivenc.jar:/opt/nfast/java/classes/
jutils.jar:/opt/nfast/java/classes/jcetools.jar:/opt/nfast/
java/classes/keysafe.jar
```

2c Save your changes.

3 Add the `novlwww` user to the `nfast` group by entering the following command:

```
usermod novlwww -G nfast
```

4 Add the netHSM certificate configuration lines to the `tomcat5.conf` file:

4a In a text editor, open the `/var/opt/novell/tomcat5/conf/tomcat5.conf` file.

4b Add the following lines:

```
JAVA_OPTS="${JAVA_OPTS} -Dcom.novell.nidp.extern.config.file=
/var/opt/novell/tomcat5/webapps/nidp/WEB-INF/classes/
externKeystore.properties"

JAVA_OPTS="${JAVA_OPTS} -Dprotect=module
-DignorePassphrase=true"
```

The first line specifies the location of the properties file. You can specify another location.

The second line is required only if you want the keystore to be module protected rather than card protected.

5 Configure the `externKeystore.properties` file to use the nCipher key and keystore:

5a In a text editor, create an `externKeystore.properties` file in the `/var/opt/novell/tomcat5/webapps/nidp/WEB-INF/classes` directory.

If you specified a different location for this file in Step 4, use that location.

5b Add the following lines:

```
com.novell.nidp.extern.signing.providerClass=com.ncipher.provider.km.
nCipherKM
com.novell.nidp.extern.signing.providerName=nCipherKM
com.novell.nidp.extern.signing.keystoreType=nCipher.sworld
com.novell.nidp.extern.signing.keystoreName=/opt/novell/devman/jcc/
certs/idp/AMstore.jks
com.novell.nidp.extern.signing.keystorePwd=mypwd
com.novell.nidp.extern.signing.alias=od93
com.novell.nidp.extern.signing.keyPwd=mypwd
```

Enter your values for the following variables:

Variable	Value
<provider_class>	The name of the providerClass. For nCipher, this must be set to com.ncipher.provider.km.nCipherKM.
<provider_name>	The name of the provider. For nCipher, this must be set to nCipherKM.
<keystore_type>	The type of keystore. For nCipher, this must be set to nCipher.sworld.
<keystore_name>	The name you specified when you created the keystore. In this sample configuration, the name is AMstore.jks.
<keystore_pwd>	When you use module-protected keys, the keystore password must be null. For example: com.novell.nidp.extern.signing.keystorePwd=
<key_alias>	The alias you created for the key when you created the key. In this sample configuration, the name is od93.
<key_pwd>	When you use module-protected keys, the key password must be null. For example: com.novell.nidp.extern.signing.keyPwd=

6 To restart Tomcat, enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

7 Continue with "Verifying the Use of the nCipher Key Pair" on page 55.

Configuring a Windows Identity Server for the Certificate

1 At the Identity Server, log in as the Windows administrator.

2 Add the nfast JAR files to the classpath.

Because the Identity Server runs as a Tomcat service, the following steps explain how to modify the classpath for Tomcat.

2a Run the `tomcat5w.exe` utility located in the following directory:

Windows Server 2003: `\Program Files\Novell\Tomcat\bin`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\bin`

2b Click the *Java* tab.

2c In the *Java Classpath* text box add the following to the end of the path:

```
";C:\nfast\java\classes\jcetools.jar;C:\nfast\java\classes\jutils.jar
;C:\nfast\java\classes\keysafe.jar;C:\nfast\java\classes\kmcsp.jar;C:
\nfast\java\classes\kmjava.jar;C:\nfast\java\classes\nfjava.jar;C:\nf
ast\java\classes\rsaprivenc.jar;C:\nfast\java\classes\spp.jar"
```

2d Save your changes.

3 Add the netHSM certificate configuration lines to the `tomcat5.conf` file:

3a Run the `tomcat5w.exe` utility located in the following directory:

Windows Server 2003: `\Program Files\Novell\Tomcat\bin`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\bin`

3b Click the *Java* tab.

3c In the *Java Options* text box, add the following as three separate lines:

```
-
Dcom.novell.nidp.extern.config.file=C:\PROGRA~1\Novell\Tomcat\webapps
   \nidp\WEB-INF\classes\externKeystore.properties
-Dprotect=module
-DignorePassphrase=true
```

The first line specifies the location of the properties file. For readability, it has been wrapped and indented. Remove the extra white space when creating the entry in the file. You can specify another location.

The second line is required only if you want the keystore to be module protected rather than card protected.

4 Configure the `externKeystore.properties` file to use the nCipher key and keystore:

4a In a text editor, create an `externKeystore.properties` file in the following directory:

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\WEB-INF\classes`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF\classes`

If you specified a different location for this file in Step 3, use that location.

4b Add the following lines:

```
com.novell.nidp.extern.signing.providerClass=com.ncipher.provider.km.
nCipherKM
com.novell.nidp.extern.signing.providerName=nCipherKM
com.novell.nidp.extern.signing.keystoreType=nCipher.sworld
com.novell.nidp.extern.signing.keystoreName=C:\\Program
Files\\Novell\\
   devman\\jcc\\certs\\idp\\AMstore.jks
com.novell.nidp.extern.signing.keystorePwd=mypwd
com.novell.nidp.extern.signing.alias=od93
com.novell.nidp.extern.signing.keyPwd=mypwd
```

The `com.novell.nidp.extern.signing.keystoreName` line is wrapped and indented for readability. All extra white space needs to be removed in the file entry. The double slashes in the path are required.

Enter your values for the following variables:

Variable	Value
`<provider_class>`	The name of the providerClass. For nCipher, this must be set to `com.ncipher.provider.km.nCipherKM`.
`<provider_name>`	The name of the provider. For nCipher, this must be set to `nCipherKM`.
`<keystore_type>`	The type of keystore. For nCipher, this must be set to `nCipher.sworld`.
`<keystore_name>`	The name you specified when you created the keystore. In this sample configuration, the name is `AMstore.jks`.

Variable	Value
<keystore_pwd>	When using module-protected keys, the keystore password must be null. For example: com.novell.nidp.extern.signing.keystorePwd=
<key_alias>	The alias you created for the key when you created the key. In this sample configuration, the name is od93.
<key_pwd>	When using module-protected keys, the key password must be null. For example: com.novell.nidp.extern.signing.keyPwd=

5 To restart Tomcat, enter the following commands:

```
net stop Tomcat5
net start Tomcat5
```

6 Continue with "Verifying the Use of the nCipher Key Pair" on page 55.

Verifying the Use of the nCipher Key Pair

After you have configured the Identity Server to use the nCipher key pair and have restarted Tomcat, the metadata of the Identity Server indicates that the nCipher key pair is being used for the signing certificate.

1 In a browser, enter the following URL:

```
http://<DNS_name>:8080/nidp/idff/metadata
```

Replace <*DNS_name*> with the DNS name of your Identity Server.

2 Search for the following string:

```
<md:KeyDescriptor use="signing">
```

3 Copy the certificate text between the <ds:X509Certificate> and the </ds:X509Certificate> tags

4 Paste the text into a text editor.

5 Delete the <ds:X509Certificate> tag and replace it with the following text:

```
-----BEGIN CERTIFICATE-----
```

6 Delete the </ds:X509Certificate> tag and replace it with the following text:

```
-----END CERTIFICATE-----
```

7 Save the file as a text file with a .cer extension.

8 Open the file in Internet Explorer.

9 View the certificate details.

If the Identity Server is using the nCipher signing certificate, the certificate is issued by your CA and the name the certificate is issued to is the name you specified for the certificate.

If the Identity Server is using the Access Manager certificate, the certificate is issued by the Organizational CA and the certificate name is test-signing. For troubleshooting information, see "Troubleshooting the netHSM Configuration" on page 56.

Troubleshooting the netHSM Configuration

To discover potential configuration errors:

1. Verify that you have not enabled the data encryption of resource IDs. There is a known issue with this feature and the Apache libraries in a multi-provider environment. Because of this issue, netHSM is not compatible with encrypting the resource IDs.

   1a In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.

   1b Click a profile, then check the setting for the *Have Discovery Encrypt This Service's Resource Ids* option.

   1c If the option is selected, deselect it, then click *OK*.

   1d Verify that all profiles have been configured so that they do not encrypt the resource IDs.

2. View the nfast log files:

   Linux: `/opt/nfast/log`

   Windows: `C:\nfast\log`

   When there is a port conflict, `logfile` contains entries similar to the following:

   ```
   nFast server: Notice: Using tcp socket local:9000
   nFast server: Fatal error during startup: Operating system call failed:
   bind tcp socket, Address already in use
   ```

   For information on how to change the port, see Step 6 on page 45. For other errors, consult the netHSM documentation.

3. (Linux only) If the `novlwww` user does not have rights to the `cmdadp.log` and `cmdadp-debug.log` files, the Identity Server is halted because it cannot read the keystore. The Health page of the Identity Server displays the following error:

   ```
   The following error occurred during the identity server configuration.
   Unable to read keystore: /opt/novell/devman/jcc/certs/idp/AMstore45.jks
   ```

   To correct the error:

   3a View the rights for the nfast log files with the following command:

   ```
   ll /opt/nfast/log
   ```

   Your listing should look similar to the following:

   ```
   -rw-r--r-- 1 novlwww nfast    0 Apr 11 11:50 cmdadp-debug.log
   -rw-r--r-- 1 novlwww nfast  134 Apr 11 11:50 cmdadp.log
   -rw-r----- 1 root    nfast   43 Apr 11 11:49 debug
   -rw-r----- 1 nfast   nfast    5 Apr 11 11:49 hardserver.pid
   -rw-r----- 1 nfast   nfast 3057 Apr 11 11:50 logfile
   ```

   If `novlwww` is not listed as the owner of the `cmdadp.log` and `cmdadp-debug.log` files, continue with Step 3b.

   If `novlwww` is listed as the owner of the files with rw permissions, log file ownership is not the source of your problem. Continue with Step 4.

   3b Stop Tomcat with the following command:

   ```
   /etc/init.d/novell-tomcat5 stop
   ```

   3c Stop nfast with the following command:

   ```
   /opt/nfast/sbin/init.d-nfast stop
   ```

3d Delete all the log files in the `/opt/nfast/log` directory.

3e Start nfast with the following command:

```
/opt/nfast/sbin/init.d-nfast start
```

3f Start Tomcat with the following command

```
/etc/init.d/novell-tomcat5 start
```

3g Wait a minute, then list the files in the `/opt/nfast/log` directory.

The nfast client creates the log files and assigns the correct owners and rights.

4 Enable Identity Server logging and view the `catalina.out` file.

4a In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Logging*.

4b Configure the following options:

File Logging: Specify enabled.

Echo to Console: Select this option.

Component File Logger Levels: Set *Application* to *debug*.

4c Click *OK*, then update the Identity Server.

4d Delete the current `catalina.out` file in the `/var/opt/novell/tomcat5/logs` directory.

4e Restart Tomcat by entering the following command:

```
/etc/init.d/novell-tomcat5 restart
```

4f To tail the `catalina.out` file, enter the following command:

```
tail -f /var/opt/novell/tomcat5/logs/catalina.out
```

4g Search for a list of providers. When nCipher is working, the file contains entries similar to the following nCipher entries:

```
Security Providers:
     SUN: 1.42
        SUN (DSA key/parameter generation; DSA signing; SHA-1, MD5
digests; SecureRandom; X.509 certificates; JKS keystore; PKIX
CertPathValidator; PKIX CertPathBuilder; LDAP, Collection CertStores)
     SunJSSE: 1.42
        Sun JSSE provider(implements RSA Signatures, PKCS12, SunX509
key/trust factories, SSLv3, TLSv1)
     SunRsaSign: 1.42
        SUN's provider for RSA signatures
     SunJCE: 1.42
        SunJCE Provider (implements DES, Triple DES, AES, Blowfish,
PBE, Diffie-Hellman, HMAC-MD5, HMAC-SHA1)
     SunJGSS: 1.0
        Sun (Kerberos v5)
     nCipherRSAPrivateEncrypt: 1.008004
        RSA private key encrypt handling provider
     nCipherKM: 1.008004
        nCipher Secure Key Management
     BC: 1.28
        BouncyCastle Security Provider v1.28
     SAML: 1.0
        SAML SASL Mechanism
```

4h (Conditional) If the `catalina.out` file does not contain any entries for providers, check for the following errors:

- Check the Health of the Identity Server. If the status is red, use the error message to resolve the issue.
- Make sure the `novlwww` user has read rights to the keystore.
- Verify that the `externKeystore.properties` file has all the required lines with valid values. See Step 5 on page 52.
- Verify that the `tomcat5.conf` file is configured correctly. See Step 4 on page 52.

5 Enable netHSM logging.

This logging feature is very verbose. It should be turned on only while you are debugging a problem. If it is left on, your machine can quickly run out of disk space.

5a To the `tomcat5.conf` file in the `/var/opt/novell/tomcat5/conf` directory, add the following line:

```
JAVA_OPTS="${JAVA_OPTS} -DJCECSP_DEBUG=255 -DJCECSP_DEBUGFILE=/var/
opt/novell/tomcat5/logs/nCipher_jcecsp.debug"
```

5b Restart Tomcat by entering the following command:

```
/etc/init.d/novell-tomcat5 restart
```

5c Look for clues in the `nCipher_jcecsp.debug` file.

Customizing Login Pages, Logout Pages, and Messages

2

2.1 Customizing the Identity Server Login Page

You can create custom login pages that are displayed when the user authenticates to the Identity Server. There are a multitude of reasons for customizing the login page. You might want to remove the Novell branding and replace it with your company’s brands. You might need to authenticate users with non-default attributes (such as an e-mail address rather than a username). You also might be fronting several protected resources with an Access Gateway, and you need to create a unique login page for each resource.

When you customize the login page:

- You need to decide on the type of page to use. See Section 2.1.1, “Selecting the Login Page and Modifying It,” on page 60.
- You need to configure the Identity Server to display the correct login page. See Section 2.1.2, “Configuring the Identity Server to Use Custom Login Pages,” on page 72.
- If the custom page doesn’t display, you need to discover the cause. See Section 2.1.3, “Troubleshooting Tips for Custom Login Pages,” on page 77.

Using Custom Pages from Previous Releases: The process for customizing login pages was modified in Access Manager 3.1 SP1. This new process requires some modifications to login pages that have been customized for either 3.1 or 3.0. If you need information on these modification procedures, see the following sections in the *Novell Access Manager 3.1 SP2 Installation Guide*:

- “Modifying 3.0 Login Pages for 3.1 SP2”
- “Upgrading from Access Manager 3.1 to 3.1 SP2”

Modifying the Target of the User Portal: If you want to control the target when users log directly into the Identity Server, see Section 3.6.2, “Specifying a Target,” on page 135.

Modifying Error Pages: Both the Identity Server and the Access Gateway return error pages to the user. For information on customizing these messages and pages, see the following:

- “Customizing Identity Server Messages” on page 80
- “Customizing Error Pages on the Access Gateway Appliance” or “Customizing the Error Pages of the Access Gateway Service” in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*

2.1.1 Selecting the Login Page and Modifying It

You must be familiar with customizing JSP files to create a customized login page. You can use any of the following methods to produce the page:

- If you only need to customize the credentials (for example, prompt the user for an e-mail address rather than a name), you can make most of the modifications in the Administration Console. You need to add some properties to a method, create a contract from that method, and modify the prompt in the `login.jsp` file. For configuration information, see "Customizing the Default Login Page to Prompt for Different Credentials" on page 61.
- If you want to maintain the features of the 3.1 page and use its authentication cards but you want to remove the Novell branding, you need to modify the `nidp.jsp` file. The `nidp.jsp` file uses iframes, so the devices that your users use for authentication must also support iframes. For configuration information, see "Customizing the nidp.jsp File" on page 64.
- If you don't need the authentication cards and if the devices that your users use for authentication support iframes, you can start with the `login.jsp` file and customize it. For configuration information, see "Modifying the 3.1 login.jsp File" on page 68.
- If some of your users are using devices that don't support iframes, you need to customize the 3.0 login page. For configuration information, see "Modifying the 3.0 Login Page" on page 69.

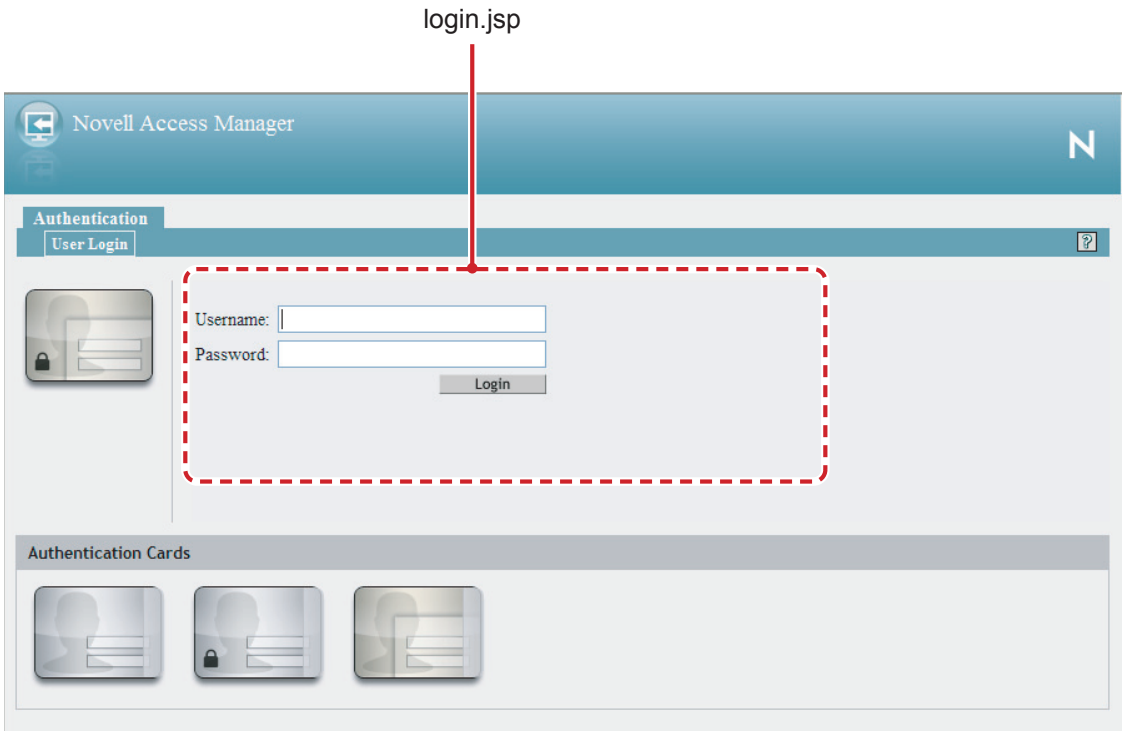
IMPORTANT: After you have created customized login pages, you need to back them up before doing an upgrade. The upgrade process overrides any custom changes made to JSP files that use the same filename as those included with the product.

During an upgrade, you can select to restore custom login pages, but Novell still recommends that you have your own backup of any customized files.

Customizing the Default Login Page to Prompt for Different Credentials

This section explains how to prompt the users for an identifier other than the user's name. Figure 2-1 displays the default login page with the username prompt.

Figure 2-1 *Modifying the Credential Prompts*



This section explains how to modify the content of the `login.jsp` file. If you want to modify other aspects of this page, you need to select one of the other methods.

The instructions below explain how to create a method that sets up the appropriate query so that the user can be found in the user store with an identifier other than the username (the cn attribute). The instructions then explain how to create a contract that uses this method and how to modify the `login.jsp` page so that it prompts for the appropriate identifier such as an email address instead of a username.

1. Create a method with the appropriate query:
   - **1a** In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Methods*.
   - **1b** Click *New*, then specify a *Display Name*.
   - **1c** In the drop-down menu for classes, select a class that is a username/password class.
   - **1d** Leave the *Identifies User* option enabled, and configure the user store option according to your needs.
   - **1e** In the *Properties* section, click *New*, then specify the following values:

     Property Name: `Query`

     Property Value: `(objectclass=person)(mail=%Ecom_User_ID%)`

This property is defined so that it queries the user store for the attribute you want to use rather than the cn attribute (in this case, the mail attribute of the person class). The `%Ecom_User_ID%` variable is the default variable name on the login page. You can change this to `%EMail_Address%` if you also change the value in your custom login page.

For more information on how to use this property, see "Query Property" on page 121.

1f In the *Properties* section, click *New*, then specify the following values:

Property Name: `JSP`

Property Value: *<filename>*

Replace *<filename>* with the name of the custom `login.jsp` page you are going to create so that the page prompts the user for an e-mail address rather than a username. This must be the filename without the JSP extension. For example, if you name your file `email_login.jsp`, then you would specify `email_login` for the property value.

1g Click *OK*.

2 Create a contract that uses this method:

2a Click *Contracts* > *New*.

2b Select the method you just created.

2c Configure the other options to fit your requirements.

For information on configuring the other options for a contract, see Section 3.4, "Configuring Authentication Contracts," on page 125.

2d Click *OK*.

3 Update the Identity Server.

4 Copy the `login.jsp` file and rename it. The JSP files are located on the Identity Server in the following directory:

Linux: `/var/opt/novell/tomcat5/webapps/nidp/jsp`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\jsp`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\jsp`

5 (Conditional) If you modified the `%Ecom_User_ID%` variable, find the string in the file and replace it with your variable.

6 (Conditional) If you need to support only one language, modify the prompt in the `login.jsp` file:

6a Find the following string in the file:

```
<label><%=handler.getResource(JSPResDesc.USERNAME)%></label>
```

6b Replace it with the string you want, for example:

```
<label>Email Address:</label>
```

6c Copy the modified file to each Identity Server in the cluster.

6d Back up your customized file.

7 (Conditional) If you need to localize the prompt for multiple languages, create a custom message properties file for the login prompt. (For more information on how to create a custom message properties file, see Section 2.3.1, "Customizing Messages," on page 80.)

The following steps assume you want to change the username prompt to an e-mail address prompt.

7a Find the following definition in the `com/novell/nidp/resource/jsp` directory of the unzipped `nidp.jar` file.

```
JSP.50=Username:
```

7b Add this definition to your custom properties file and modify it so that it prompts the user for an e-mail address.

```
JSP.50=Email Address:
```

7c Translate the value and add this entry to your localized custom properties files.

7d Copy the customized properties files to the `WEB-INF/classes` directory of each Identity Server in the cluster.

7e Restart Tomcat on each Identity Server.

Linux Identity Server: Enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

Windows Identity Server: Enter the following commands:

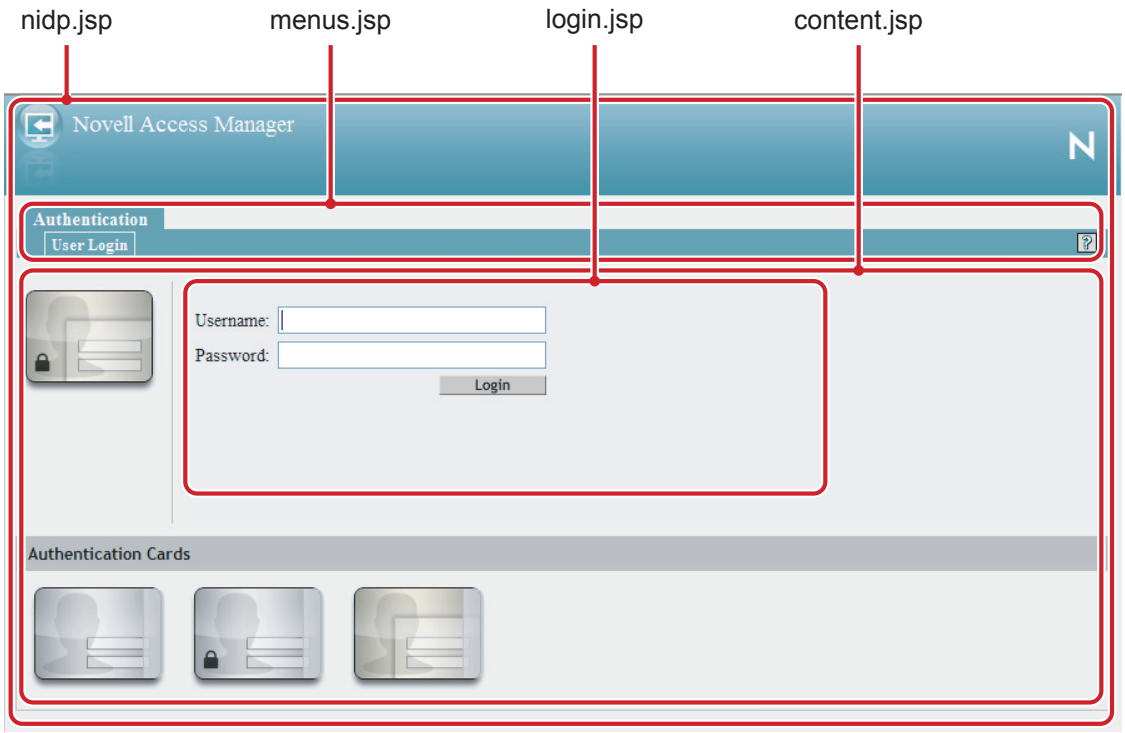
```
net stop Tomcat5
net start Tomcat5
```

8 To view a sample custom page with these modifications, see Section 2.4.1, “Modified login.jsp File for Credential Prompts,” on page 85.

Customizing the nidp.jsp File

Figure 2-2 displays the default login page provided by Access Manager. Multiple JSPs are used to create the page.

Figure 2-2 *The JSPs That Create the Login Page*



You can use the `nidp.jsp` file to customize the header with the Novell Access Manager product name and the Novell logo. The `menus.jsp` file controls the *Authentication* and *User Login* tabs. The `login.jsp` file controls the credential frame with username and password. The `content.jsp` file controls what is displayed on the page, including the available authentication cards.

The following sections explain how to modify the login page that these JSPs create:

- "Rebranding the Header" on page 64
- "Customizing the Card Display" on page 66
- "Customizing the Credential Frame" on page 66

Rebranding the Header

1. Copy the `nidp.jsp` file and rename it. The JSP files are located on the Identity Server in the following directory:

   Linux: `/var/opt/novell/tomcat5/webapps/nidp/jsp`

   Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\jsp`

   Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\jsp`

2. Replace the header title that appears in the top frame ("Novell Access Manager" in Figure 2-2):

   2a Locate the following string at the top of the file.

   ```
   String hdrTitle = handler.getResource(JSPResDesc.PRODUCT);
   ```

2b Replace the value with the title you want to appear. For example:

```
String hdrTitle = "My Company"
```

Make sure to enclose your title value with double quotes.

3 Replace the window title that appears in the browser title bar:

3a Locate the following line that appears between the `<head></head>` tags:

```
<title><%=handler.getResource(JSPResDesc.TITLE)%></title>
```

3b Replace the content between the `<title>` and `</title>` tags with the title you want to appear. For example:

```
<title>My Company</title>
```

4 Replace the Access Manager logo on the left of the header (see Figure 2-2):

4a Locate the following string:

```
String hdrImage = "AMHeader_image.png";
```

4b Replace the value in the quotes with the path and the filename of the image you want to use.

For example, if you created a `/custom_images` directory in the `images` directory, the `hdrImage` string would have a value similar to the following:

```
String hdrImage = "/custom_images/myapp.png"
```

5 Replace the Novell logo on the right of the header (see Figure 2-2):

5a Locate the following string:

```
String hdrLogo = "AMHeader_logo.png";
```

5b Replace the value of the `hdrLogo` string with the path and the filename of the image you want to use.

For example, if you created a `/custom_`images directory in the `images` directory, the `hdrLogo` string would have a value similar to the following:

```
String hdrLogo = "/custom_images/companylogo.png"
```

6 To change the background image for the header (which allows for variable sizing of the page):

6a Locate the following string:

```
String hdrBgndImg = "AMHeader_background.png";
```

6b Replace the value of the `hdrBgndImg` string with the path and the filename of the image you want to use. You can use a color or an image that can be repeated. The style is set to repeat it from left to right as the window expands.

For example, if you created a `/custom_images` directory in the `images` directory, the `hdrBgndImg` string would have a value similar to the following:

```
String hdrBgndImg = "/custom_images/mybackground.png"
```

7 If your custom images or title do not appear in the header where you want them, you need to modify the style section.

7a Locate the following lines:

```
#header { background-image: url(<%=
handler.getImage(hdrBgndImg,false)%>); background-repeat: repeat-x; }

#logo { position: absolute; top: 0px; right: 0px; }

#title { position: absolute; font-size: 1.2em; color: white; top:
13px; left: 55px; }
```

7b Modify the top, left, and right values.

8 To change the background colors on the page, modify the color values in the `<style>` section of the `<head>` element.

9 If you need to create multiple custom login pages, repeat Step 1 through Step 8.

10 Copy the custom login pages and the images they require to each Identity Server in the cluster.

11 Continue with one of the following tasks:

- To modify what appears in the credential frame, continue with "Customizing the Credential Frame" on page 66.
- To control the cards displayed in the Authentication Cards section, see "Customizing the Card Display" on page 66.
- To configure the Identity Server to use your custom pages, see "Adding Logic to the main.jsp File" on page 73.
- To view a sample custom page with these modifications, see Section 2.4.2, "Custom nidp.jsp File with Custom Credentials," on page 88.

Customizing the Card Display

The easiest method to control what appears in the *Authentication Cards* section is not by modifying the `content.jsp` file. It is by using the *Show Card* option that appears on the definition of each card. If this option is not selected, the card does not appear in the *Authentication Cards* section. Each contract has an associated card. For information on modifying the card options, see Section 3.4, "Configuring Authentication Contracts," on page 125.

Continue with one of the following:

- To modify what appears in the credential frame, continue with "Customizing the Credential Frame" on page 66
- To configure the Identity Server to use your custom pages, see "Adding Logic to the main.jsp File" on page 73.

Customizing the Credential Frame

The most common reason for modifying the `login.jsp` page is to prompt the users for an identifier other than the user's name. To do this, you need to create a method that sets up the appropriate query so that the user can be found in the user store with an identifier other than the username. You then need to create a contract that uses this method. You also need to modify the prompt in the `login.jsp` page to match the identifier you are prompting for.

1 Create a method with the appropriate query:

1a In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Methods*.

1b Click *New*, then specify a *Display Name*.

1c In the drop-down menu for classes, select a class that is a username/password class.

1d Leave the *Identifies User* option enabled, and configure the user store option according to your needs.

1e In the *Properties* section, click *New*, then specify the following values:

Property Name: `Query`

Property Value: `(objectclass=person)(mail=%Ecom_User_ID%)`

This property is defined so that it queries the user store for the attribute you want to use rather than the cn attribute (in this case, the mail attribute of the person class). Change `mail` to the name of the attribute in your user store that you want to use for the user identifier.

The `%Ecom_User_ID%` variable is the default variable name on the login page. You can change this to something like `%EMail_Address%` if you also change the value in your custom login page.

For more information on how to use this property, see "Query Property" on page 121.

1f In the *Properties* section, click *New*, then specify the following values:

Property Name: `JSP`

Property Value: *<filename>*

Replace <*filename*> with the name of the custom `login.jsp` page you are going to create so that the page prompts the user for an e-mail address rather than a username. This must be the filename without the JSP extension. For example, if you name your file `email_login.jsp`, then you would specify `email_login` for the property value.

1g Click *OK*.

2 Create a contract that uses this method:

2a Click *Contracts* > *New*.

2b Select the method you just created.

2c Configure the other options to fit your requirements.

If you are creating multiple custom login pages with customized credentials, you might want to use the URI to hint at which custom `login.jsp` file is used with which custom `nidp.jsp` file. For example, the following URI values have the filename of the login page followed by the name of the custom `nidp.jsp` page:

```
login1/custom1
login2/custom2
login3/custom3
```

For information on configuring the other options for a contract, see Section 3.4, "Configuring Authentication Contracts," on page 125.

2d Update the Identity Server.

3 Copy the `login.jsp` file and rename it. The JSP files are located on the Identity Server in the following directory:

Linux: `/var/opt/novell/tomcat5/webapps/nidp/jsp`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\jsp`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\jsp`

4 (Conditional) If you modified the `%Ecom_User_ID%` variable, find the string in the file and replace it with your variable.

5 (Conditional) If you need to support only one language, modify the prompt in the `login.jsp` file:

5a Find the following string in the file:

```
<label><%=handler.getResource(JSPResDesc.USERNAME)%></label>
```

5b Replace it with the string you want, for example:

```
<label>Email Address:</label>
```

5c Copy the modified file to each Identity Server in the cluster.

5d Back up your customized file.

6 (Conditional) If you need to localize the prompt for multiple languages, create a custom message properties file for the login prompt. (For more information on how to create a custom message properties file, see Section 2.3.1, "Customizing Messages," on page 80.)

The following steps assume you want to change the username prompt to an e-mail address prompt.

6a Find the following definition in the `com/novell/nidp/resource/jsp` directory of the unzipped `nidp.jar` file.

```
JSP.50=Username:
```

6b Add this definition to your custom properties file and modify it so that it prompts the user for an e-mail address.

```
JSP.50=Email Address:
```

6c Translate the value and add this entry to your localized custom properties files.

6d Copy the customized properties files to the `WEB-INF/classes` directory of each Identity Server in the cluster.

6e Restart Tomcat on each Identity Server.

Linux Identity Server: Enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

Windows Identity Server: Enter the following commands:

```
net stop Tomcat5
net start Tomcat5
```

7 To view a sample custom page with these modifications, see Section 2.4.2, "Custom nidp.jsp File with Custom Credentials," on page 88.

8 To specify which customized `nidp.jsp` to display with the contract, you must modify the `main.jsp` file. Continue with "Adding Logic to the main.jsp File" on page 73.

Modifying the 3.1 login.jsp File

The `login.jsp` file gives you just the credential frame with the login prompts in an iframe. It has no branding header. If you use this page, you are responsible for writing the HTML code for the header and the branding.

1 Copy the `login.jsp` file and rename it. The JSP files are located on the Identity Server in the following directory:

Linux: `/var/opt/novell/tomcat5/webapps/nidp/jsp`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\jsp`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\jsp`

2 Add the custom branding and any other content you require to the file.

3 To modify the credentials, see "Customizing the Credential Frame" on page 66.

4 Repeat Step 1 through Step 3 for each resource that requires unique branding.

5 Copy the files to each Identity Server in the cluster.

6 Back up your customized files.

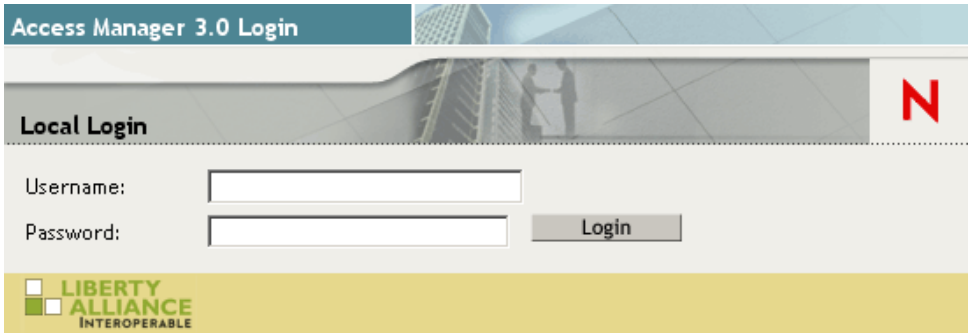
7 (Optional) To view a sample custom page with these modifications, see Section 2.4.3, "Custom 3.1 login.jsp File," on page 95.

8 Continue with "Using Properties to Specify the Login Page" on page 72.

Modifying the 3.0 Login Page

If you need a login page that doesn't use iframes, you can use the 3.0 login page as the starting file for your custom login page. Figure 2-3 illustrates the default look and feel of this page.

Figure 2-3 *Access Manager 3.0 Default Login Page*



You can change the Novell branding and modify the credential prompts.

- "Modifying the Branding in the 3.0 Login Page" on page 69
- "Modifying the Credentials in the 3.0 Login Page" on page 70

Modifying the Branding in the 3.0 Login Page

1 Copy the `/var/opt/novell/tomcat4/webapps/nidp/jsp/login.jsp` file from your 3.0 Identity Server and rename it.

If you do not have a 3.0 `login.jsp` file, copy the modified version of this file from "Modifications Required for a 3.0 Login Page" in the *Novell Access Manager 3.1 SP2 Installation Guide* to a true text editor. Delete all the extra line breaks.

2 (Conditional) If you are using the file from your 3.0 Identity Server, modify it so that it can compile on a 3.1 Identity Server. For instructions, see "Modifications Required for a 3.0 Login Page" in the *Novell Access Manager 3.1 SP2 Installation Guide*.

3 Replace the Access Manager 3.0 Login string:

3a Find the following line in the file:

```
<div id="title"><b><%=handler.getResource(JSPResDesc.TITLE)%></b></
div>
```

3b Replace `<%=handler.getResource(JSPResDesc.TITLE)%>` with your string. Your line should look similar to the following:

```
<div id="title"><b>HHB Partner</b></div>
```

4 Replace the Local Login string:

When a 3.0 page runs on a 3.1 system, the Local Login string is replaced by the product string, "Novell Access Manager". To modify this string:

4a Locate the following string in the file.

```
<div id="locallabel"><b><%=handler.getResource(JSPResDesc.PRODUCT)%></
b></div>
```

4b Replace `<%=handler.getResource(JSPResDesc.PRODUCT)%>` with the title you want to appear. For example:

```
<div id="locallabel"><b>My Company</b></div>
```

5 Replace the window title that appears in the browser title bar:

5a Find the following lines in the file:

```
<META HTTP-EQUIV="Content-Language" CONTENT="<%=handler.getLanguage
Code()%>">
<title><%=handler.getResource(JSPResDesc.TITLE)%></title>
```

5b Replace the content between the `<title>` and `</title>` tags with the title you want to appear. For example:

```
<title>My World</title>
```

6 Remove the Novell N logo:

6a Find the following line in the file:.

```
<div id="headimage"><img src="<%= request.getContextPath() %>/images/
Odyssey_LoginHead.gif" alt="" height="80" width="550" border="0"></
div>
```

6b Replace `Odyssey_LoginHead.gif` with `Odyssey_Head.gif`.

6c Save the file.

7 Select one of the following tasks:

- To modify what appears in the credential frame, continue with "Modifying the Credentials in the 3.0 Login Page" on page 70.
- To view a file with these modifications, see Section 2.4.4, "Custom 3.0 login.jsp File," on page 98.
- To configure the Identity Server to use your custom pages, see "Using Properties to Specify the Login Page" on page 72.

Modifying the Credentials in the 3.0 Login Page

1 Create a method with the appropriate query:

1a In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Methods*.

1b Click *New*, then specify a *Display Name*.

1c In the drop-down menu for classes, select a class that is a username/password class.

1d Leave the *Identifies User* option enabled, and configure the user store option according to your needs.

1e In the *Properties* section, click *New*, then specify the following values:

Property Name: `Query`

Property Value: `(objectclass=person)(mail=%Ecom_User_ID%)`

This property is defined so that it queries the user store for the attribute you want to use rather than the cn attribute (in this case the mail attribute of the person class). The `%Ecom_User_ID%` variable is the default variable name on the login page. You can change this to `%EMail_Address%` as long as you also change the value in your custom login page.

For more information on how to use this property, see "Query Property" on page 121.

1f Click *OK*.

1g Create a contract that uses this method.

For information on configuring a contract, see Section 3.4, "Configuring Authentication Contracts," on page 125.

1h Update the Identity Server.

2 (Conditional) If you need to support only one language, modify the string in your custom login file:

2a Find the following string in the file:

```
<label style="width: 100px"><%=handler.getResource(JSPResDesc.
USERNAME)%></label>
```

2b Replace it with the string you want, for example:

```
<label style="width: 100px">Email Address:</label>
```

2c Copy the modified file to each Identity Server in the cluster.

2d Update the Identity Server cluster.

2e Back up your customized file.

3 (Conditional) If you need to localize the prompt for multiple languages, create a custom message properties file for the login prompt. (For more information on how to create a custom message properties file, see Section 2.3.1, "Customizing Messages," on page 80.)

The following steps assume you want to change the Username prompt to an Email Address prompt.

3a Find the following definition in the `com/novell/nidp/resource/jsp` directory of the unzipped `nidp.jar` file.

```
JSP.50=Username:
```

3b Add this definition to your custom properties file and modify it so that it prompts the user for an e-mail address.

```
JSP.50=Email Address:
```

3c Translate the value and add this entry to your localized custom properties files.

3d Copy the customized properties files to the `WEB-INF/classes` directory of each Identity Server in the cluster.

3e Copy the custom login page to the JSP directory of each Identity Server in the cluster.

3f Restart Tomcat on each Identity Server.

Linux Identity Server: Enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

Windows Identity Server: Enter the following commands:

```
net stop Tomcat5
net start Tomcat5
```

4 (Optional) To view a customized 3.0 login page, see Section 2.4.4, "Custom 3.0 login.jsp File," on page 98.

5 Continue with "Using Properties to Specify the Login Page" on page 72.

2.1.2 Configuring the Identity Server to Use Custom Login Pages

There are two ways to configure the Identity Server to use a custom login page. You can use properties or you can modify the `main.jsp` file. Which method you can use depends upon your modifications.

- You can use properties if you created your custom page from the 3.1 `login.jsp` page or have modified a 3.0 custom page to work on 3.1. See "Using Properties to Specify the Login Page" on page 72.
- If you created your custom page from the `nidp.jsp` file, you cannot use properties to specify the main custom page for authentication. You must modify the `main.jsp` file. See "Adding Logic to the main.jsp File" on page 73.

Using Properties to Specify the Login Page

For each resource that needs a unique login page, you need to create an authentication method and add the JSP and MainJSP properties to the method. You then need to create a contract for each method.

The following steps assume that the custom login page is called `custom1.jsp`.

1 Create a method for a custom login page:

1a In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Methods*.

1b Select one of the following actions:

- If you have create a method for a Query property to be used with your custom login page, click the name of the method.
- If you didn't modify the credentials on the login page, click *New*, specify a display name, select a password class, and configure a user store.

1c In the *Properties* section, click *New*, then specify the following:

Property Name: `MainJSP`

Property Value: `true`

This property indicates that you want to use a custom login page with this method. It also indicates that the custom login page contains the prompts for user credentials.

Property names and values are case sensitive.

1d Click *OK*.

1e (Conditional) If the *Properties* section does not contain a JSP property, click *New*, specify the following:

Property Name: `JSP`

Property Value: `custom1`

The property value for the JSP property is the name of the custom login file without the `JSP` extension. Replace `custom1` with the name of your custom login file. This property determines which login page is displayed when this method is used. The filename cannot contain `nidp` as part of its name.

1f Click *OK*.

For more information about setting property values, see Section 3.2.2, "Specifying Common Class Properties," on page 121.

1g (Conditional) If you created multiple custom login pages, repeat Step 1b through Step 1e for each page.

2 For each method that you modified for a custom login page, create a contract:

2a Click *Contracts*, then click *New*.

2b Fill in the fields to fit the needs of the resource, but make sure to assign the custom method as the method for the contract.

2c Click *Next*, configure a card for the contract, then click *Finish*.

3 Update the Identity Server.

4 For each resource that you have created a custom login page, assign that resource to use the contract that is configured to display the appropriate login page:

4a Click *Devices* > *Access Gateways* > *Edit* > *[Reverse Proxy Name]* > *[Proxy Service Name]* > *Protected Resources*.

4b For each protected resource that you have created a custom contract for, select the protected resource, then configure it to use the custom contract.

5 Update the Access Gateway.

6 (Conditional) If the custom page does not display correctly, see Section 2.1.3, "Troubleshooting Tips for Custom Login Pages," on page 77.

Adding Logic to the main.jsp File

You can modify the `main.jsp` file and use the contract URI to specify the login page to display. The Identity Server must be running 3.1 SP1 or later to use this feature. Be aware of the following:

- The `main.jsp` file cannot be renamed, so any modifications you make to this file can be lost whenever you upgrade the Identity Server. During the upgrade, you must select to restore custom files or you must restore your modified file after the upgrade. If this is the only JSP file that you modified that uses an Identity Server name, it is probably best to manually restore this file after an upgrade.
- Modifying the `main.jsp` file requires knowledge of JSP programming and if/else statements.

Modifying the `main.jsp` file allows you to have the following type of configuration:

- You can create multiple customized `nidp.jsp` pages. For example: `custom1.jsp`, `custom2.jsp`, and `custom3.jsp`.
- You can create multiple customized `login.jsp` pages that request different login credentials. For example:

  login1.jsp: Configured to request username and password.

  login2.jsp: Configured to request username, email, and password.

login3.jsp: Configured to request email and password.

With this type of configuration, you must create three different authentication contracts with an authentication method with a JSP property defined for each of them. These contracts require the types of values listed in the table below. The URI is defined so that it reflects the custom `login.jsp` and the custom `nidp.jps` that are used by the contract.

Contract	Configuration Details	
Contract1	URI	login1/custom1
	Method1	Configured with the following JSP property: **Property Name:** `JSP` **Property Value:** `login1` This method does not need a query property unless you are using an attribute other than the cn attribute for the username.
Contract2	URI	login2/custom2
	Method2	Configured with the following two properties: **Property Name:** `JSP` **Property Value:** `login2` **Property Name:** `Query` **Property Value:** `(&(objectclass=person)(mail=%Ecom_User_ID%))`
Contract3	URI	login3/custom3
	Method3	Configured with the following two properties: **Property Name:** `JSP` **Property Value:** `login3` **Property Name:** `Query` **Property Value:** `(objectclass=person)(mail=%Ecom_User_ID%)`

The following procedure explains how to configure Access Manager to display these custom login pages with custom credentials.

1. Create a unique method for each custom `login.jps` file:
   - **1a** In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Methods*.
   - **1b** Click *New*, then configure the following fields:

     Display name: Specify a name for the method. You might want to use a name that indicates which login page is assigned to this method.

     Class: Select a name/password class.

     Configure the other fields to match your requirements.
   - **1c** In the *Properties* section, add a Query property if the page uses custom credentials.

For example, to add an email address to the login prompts, add the following property:

Property Name: `Query`

Property Value: `(&(objectclass=person)(mail=%Ecom_User_ID%))`

If you are creating a method for Contract 1 in the example above (which prompts for a username and password), you do not need to add a query property unless you are using an attribute other than the cn attribute for the username.

1d In the Properties section, add a JSP property to specify which `login.jsp` file to use with this method.

For example:

Property Name: `JSP`

Property Value: `login2`

1e Click *Finish*.

1f If you have created more than one custom `login.jsp` file, repeat Step 1b through Step 1e for each page.

To configure the scenario described in this section, repeat these steps for three login pages.

2 Create a unique contract URI:

2a In the Administration Console, click *Contracts*.

2b Click *New*, then configure the following fields:

Display name: Specify a name for the contract. You might want to use a name that indicates which login page is assigned to this contract.

URI: Specify a value that uniquely identifies the contract from all other contracts. No spaces can exist in the URI field. You might want to use a name that indicates the custom login page and custom credential page, such as `login1/custom1`.

Methods and Available Methods: Select the authentication method you configured in Step 1.

2c Configure the other fields to meet your network requirements, then click *Next*.

2d Configure the authentication card, then click *Finish*.

2e (Conditional) If you have created multiple custom login pages, repeat Step 2b through Step 2d for each page.

To configure the scenario described in this section, repeat these steps for /login2/custom2 and /login3/custom3.

2f Click *OK*, then update the Identity Server.

3 Modify the `main.jsp` file:

3a Open the `main.jsp` file. The file is located in the following directory:

Linux: `/var/opt/novell/tomcat5/webapps/nidp/jsp`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\jsp`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\jsp`

3b Near the top of the file, add the following line:

```
String strContractURI = hand.getContractURI();
```

This sets the `strContractURI` variable to the value of the contract URI that is being used for authentication. These lines should look similar to the following:

```
<%
    ContentHandler hand = new ContentHandler(request,response);
    String strContractURI = hand.getContractURI();

    // Is there a JSP defined on a class definition or a method
    // definition that should be displayed as the main jsp here?
    if (handler.contractDefinesMainJSP())
    {
%>
```

3c After the `if` statement, add an `else if` statement for each contract URI you have created. For example:

```
else if(strContractURI != null && strContractURI.equals("login1/
custom1"))
    {
%>
     <%@ include file="custom1.jsp" %>

<%  }
else if(strContractURI != null && strContractURI.equals("login2/
custom2"))
    {
%>
        <%@ include file="custom2.jsp" %>

else if(strContractURI != null && strContractURI.equals("login3/
custom3"))
    {
%>
        <%@ include file="custom3.jsp" %>
```

These `else if` statements set up three contracts for customized login pages:

- The first `else if` statement specifies the URI of the login1 contract and configures it to display the `custom1.jsp` page for authentication.
- The second `else if` statement specifies the URI of the login2 contract and configures it to display the `custom2.jsp` page for authentication.
- The third `else if` statement specifies the URI of the login3 contract and configures it to display the `custom3.jsp` page for authentication.

Your file should look similar to the following:

```
<%@ page language="java" %>
<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>
<%@ page import="com.novell.nidp.*" %>
<%@ page import="com.novell.nidp.resource.jsp.*" %>
<%@ page import="com.novell.nidp.ui.*" %>
<%@ page import="com.novell.nidp.common.util.*" %>
<%@ page import="com.novell.nidp.liberty.wsf.idsis.apservice.schema.*"
%>

<%
    ContentHandler hand = new ContentHandler(request,response);
    String strContractURI = hand.getContractURI();

    // Is there a JSP defined on a class definition
```

```
    // or a method definition that should be displayed
    // as the main jsp here?
    if (hand.contractDefinesMainJSP())
    {
%>
        <%@ include file="mainRedirect.jsp" %>
<%  }
    else if(strContractURI != null && strContractURI.equals("login1/
custom1"))
    {
%>
     <%@ include file="custom1.jsp" %>

<%  }
else if(strContractURI != null && strContractURI.equals("login2/
custom2"))
    {
%>
        <%@ include file="custom2.jsp" %>

else if(strContractURI != null && strContractURI.equals("login3/
custom3"))
    {
%>
        <%@ include file="custom3.jsp" %>

<%  }    // This is the jsp used by default
    else
    {
%>
        <%@ include file="nidp.jsp" %>
<%  }  %>
```

3d Copy the modified `main.jsp` file to each Identity Server in your cluster.

4 Back up your customized files.

5 For each resource that you have created a custom login page for, assign that resource to use the contract that is configured to display the appropriate login page:

5a Click *Devices* > *Access Gateways* > *Edit* > *[Reverse Proxy Name]* > *[Proxy Service Name]* > *Protected Resources*.

5b For each protected resource that you have created a custom contract for, select the protected resource, then configure it to use the custom contract.

5c Update the Access Gateway.

6 (Conditional) If the custom page does not display correctly, see Section 2.1.3, “Troubleshooting Tips for Custom Login Pages,” on page 77.

2.1.3 Troubleshooting Tips for Custom Login Pages

If your custom login page does not display or generates an error message, use the following procedure to discover the root cause:

1 Set the *Application* option of *Component File Logger Levels* to debug, update the Identity Server, attempt to log in, then view the log file.

Check for Unable to Compile errors in the log file. If your custom page does not compile, a blank page is displayed.

2 If you receive an Unable to Find File error, verify the value of the JSP property. Make sure that the value does not contain the JSP extension as part of the filename.

3 If you see pages that you have deleted or pages where your modifications have not been implemented:

3a Open the new custom file with a text editor to ensure it has a newer date than the compiled file.

If this does not solve the problem, continue with Step 3b.

3b Delete the `nidp` directory in the Tomcat work directory on each Identity Server. This forces a recompile the JSP pages.

Linux: `/var/opt/novell/tomcat5/work/Catalina/localhosts/nidp`

Windows Server 2003: `\Program Files\Novell\Tomcat\work\Catalina \localhosts\nidp`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\work\Catalina \localhosts\nidp`

3c Restart Tomcat on each Identity Server.

2.2 Customizing the Identity Server Logout

You can also use the following methods to modify the Identity Server logout page:

- Section 2.2.1, "Rebranding the Logout Page," on page 78
- Section 2.2.2, "Replacing the Logout Page with a Custom Page," on page 78
- Section 2.2.3, "Configuring for Local Rather Than Global Logout," on page 79

To customize the logout page when the user logs out of an Access Gateway protected resource, see "Customizing Logout Requests" in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*.

2.2.1 Rebranding the Logout Page

The branding in the header of the logout page is controlled by the branding of the `nidp.jsp` file. If you have modified this file for a customized login, the same branding appears in the logout page. For information on how to modify `nidp.jsp` for logos, titles, and colors, see "Rebranding the Header" on page 64.

IMPORTANT: Save a copy of your modified `nipd.jsp` file. Every time you upgrade your Identity Server, you need to restore this file.

2.2.2 Replacing the Logout Page with a Custom Page

You can create your own logout page and configure the Identity Server to use it. To do this, you need to modify the `logoutSuccess.jsp` file on the Identity Server. It is located in the following directory:

Linux: `/var/opt/novell/tomcat5/webapps/nidp/jsp`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\jsp`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\jsp`

The `logoutSuccess.jsp` file is called in a frame from the `nidp.jsp` file. You can modify the file to display what you want or you can modify it to redirect the user to your custom page. One way to provide redirection is to replace the information in the <body> element of the file with something similar to the following:

```
<body>
      <script language="JavaScript">
        top.location.href='http://<hostname/path>';
      </script>
</body>
```

Replace the <*hostname/path*> string with the location of your customized logout page.

IMPORTANT: Save a copy of your modified `logoutSuccess.jsp` file. Every time you upgrade your Identity Server, you will need to restore this file.

2.2.3 Configuring for Local Rather Than Global Logout

By default, when the Identity Server receives a logout request, the Identity Server logs the user out of any identity providers and service providers to which the user has authenticated. If you want to modify this behavior so that the logout request logs the user out of just the Identity Server and leaves the user authenticated to identity providers and service providers, you need to add the following query string to the logout URL:

```
?local=true
```

The logout URL has the following format:

```
<Base_URL>/app/logout
```

Replace <*Base_URL*> with the base URL of your Identity Server. If the base URL of your Identity Server was `hhb1.provo.novell.com:8443`, your local logout URL would be the following:

```
https://hhb1.provo.novell.com:8443/app/logout?local=true
```

To modify the `logout.jsp` file so that it performs a local logout:

1. At the Identity Server, open the logout.jsp file.

   Linux: `/var/opt/novell/tomcat5/webapps/nidp/jsp`

   Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\jsp`

   Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\jsp`

2. Find the following line in the file:

   ```
   <form method="post" target="_top" action="<%= request.getContextPath() %>/
   app/logout">
   ```

3. To the `/app/logout` string, add `?local=true`. This modified line should look similar to the following:

   ```
   <form method="post" target="_top" action="<%= request.getContextPath() %>/
   app/logout?local=true">
   ```

4. Save the file.

5 Copy the file to each Identity Server in the cluster.

6 Back up this file.

2.3 Customizing Identity Server Messages

2.3.1 Customizing Messages

1 To customize the error pages, determine whether you need one custom file or multiple files:

- If you do not need to support multiple languages, you can create one custom file for all your customized messages.
- If you need to support multiple languages, you need to create a custom file for each language you want to customize.

2 Create the custom properties file and name it:

To support one language, name the file `nidp_custom_resources.properties`.

To support multiple languages, create a `nidp_custom_resources.<le_cy>.properties` file for each supported language. Replace <*le_cy*> with the standard convention for Java Resource Bundles for the language or the language and country. For example:

```
nidp_custom_resources_en_US.properties
nidp_custom_resources_fr.properties
nidp_custom_resources_es.properties
```

If you want to support a custom messages for a language and a country and for just the language, you must create two files. For example:

```
nidp_custom_resources_es_VE.properties
nidp_custom_resources_es.properties
```

3 Copy the `nidp.jar` file to a working area. This file is located in the following directory:

Linux: `/var/opt/novell/tomcat5/webapps/nidp/WEB-INF/lib`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\WEB-INF\lib`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF\lib`

4 Unzip the `nidp.jar` file in your working directory.

5 In your working directory, locate the `.properties` files in the following directories.

```
com/novell/nidp/resource/strings
com/novell/nidp/resource/logging
com/novell/nidp/resource/jsp
com/novell/nidp/resource/jcc
com/novell/nidp/resource/noxlate
com/novell/nidp/liberty/wsf/idsis/ppservice/model
com/novell/nidp/liberty/wsf/idsis/epservice/model
com/novell/nidp/liberty/wsf/idsis/opservice/model
com/novell/nidp/liberty/wsf/idsis/apservice/model
com/novell/nidp/liberty/wsf/interaction
```

```
com/novell/nidp/liberty/wsf/idsis/ssservice/model
com/novell/nidp/servlets/handler/identityeditor
com/novell/nidp/servlets/handler/identityaccesseditor
com/novell/nidp/liberty/wsf/idsis/model
com/novell/nidp/liberty/wsf/idsis/authority/ldap/attribute/plugins/
resources
com/novell/nidp/liberty/wsf/idsis/ldapservice/model
```

The properties files that have been localized contain the messages that end users might see. The properties files that have not been localized contain messages that the end users should not see.

6 Locate the messages you want to customize and copy them to your custom file.

All the messages you want to customize are placed in this file, even though they come from different properties files. Your file should look similar to the following if you selected to customize messages from the `nidp_resources_en_US.properties` file and the `SSModelResources_en_US.properties` file. For example:

```
NIDPMAIN.100=An Identity Provider response was received that failed to
authenticate this session.
NIDPMAIN.101=A request for identity federation could not be completed.
NIDPMAIN.102=A request for identity federation termination could not be
completed.

SS.WKSLdapCreds = LDAP Credentials
SS.WKSELdapCredsUserName = LDAP User Name
SS.WKSELdapCredsUserDN = LDAP User DN
SS.WKSELdapCredsUserPassword = LDAP Password
SS.WKSX509Creds = X509 Credentials
```

7 (Conditional) If you are supporting multiple languages, copy the messages to each custom language file.

8 Replace the messages in the file with your custom messages.

Replace the string after the equals (=) sign with your translated or customized message.

If you are using double-byte characters, the characters need to be in Unicode, hexadecimal format with a `\u` prefix. For example: `\u5c71`.

9 Save the file.

10 Copy the custom properties file to the following directory on all Identity Servers in the cluster:

Linux: `/var/opt/novell/tomcat5/webapps/nidp/WEB-INF/classes`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\WEB-INF\classes`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF\classes`

11 (Optional) To enable messages about the loading of the custom properties files, enable debug logging:

11a In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Logging*.

11b In the *Component File Logger Levels* section, select *Debug* level for *Application*.

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

12 Restart Tomcat.

- **Linux Identity Server:** Enter the following command:

  ```
  /etc/init.d/novell-tomcat5 restart
  ```

- **Windows Identity Server:** Enter the following commands:

  ```
  net stop Tomcat5

  net start Tomcat5
  ```

13 (Optional) To verify the loading of the custom properties files:

13a View the log file by clicking *Auditing* > *General Logging*.

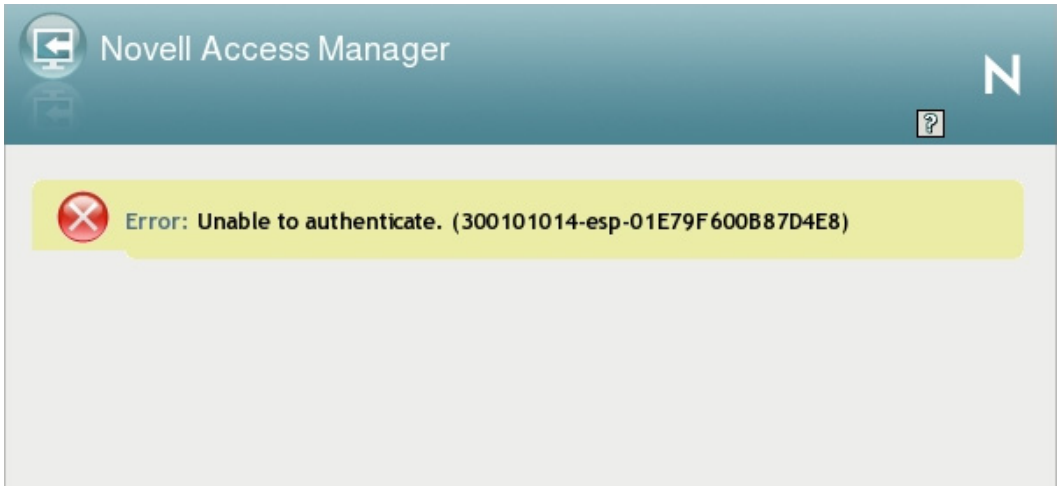
13b Search for messages similar to the following in the `catalina.out` or `stdout.log` file:

```
The named Custom Properties File was loaded and will be used:

Custom Properties File successfully loaded! Name: <Custom Properties
FileName>

An error occurred loading a specific Custom Properties File. Loading
of other Custom Properties Files will continue.

<Error Description>, Attempting to load Custom Properties File! Name:
<Custom Properties FileName>

The locale specifier in the Custom Properties File filename could not
be successfully parsed into a valid locale. Loading of other Custom
Properties Files will continue.

Custom Properties File load failed. Could not determine correct
locale! Name: <Custom Properties FileName>

A general error occurred loading Custom Properties Files. Loading will
stop and all un-loaded Custom Properties Files will not be loaded.

<Error Description>, Attempting to load Custom Properties Files!
```

To create custom error pages for the Access Gateway, see "Customizing Error Pages on the Access Gateway Appliance" in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*.

2.3.2 Customizing the Branding of the Error Page

The following page (`err.jsp`) is returned when the Identity Server encounters an error:



The file is located in the following directory.

Linux: `/var/opt/novell/tomcat5/webapps/nidp/jsp`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\jsp`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\jsp`

IMPORTANT: After you have customized this page, you need to ensure you back up this page before doing an upgrade. The upgrade process overrides any custom changes made to the `err.jsp` page.

For information on customizing the error message, see Section 2.3.1, "Customizing Messages," on page 80.

You can customize the following items:

- The window title and the display title. See "Customizing the Titles" on page 83.
- The header image and the Novell logo. See "Customizing the Images" on page 83.
- Background colors. See "Customizing the Colors" on page 84.

Customizing the Titles

The window title appears in the browser title bar. To replace this text, open the `err.jsp` file and locate the following text that appears between the `<head></head>` tags:

```
<title><%=handler.getResource(JSPResDesc.TITLE)%></title>
```

Replace the content between the `<title>` and `</title>` tags with the title you want to appear. For example:

```
<title>My Company</title>
```

The display title is the title that appears in the top frame of the page. Locate the following text that appears in the `<body>` of the page:

```
<div id="title"><%=handler.getResource(JSPResDesc.PRODUCT)%></div>
```

Replace the content between the `<div id="title">` and `</div>` with the title you want to appear. For example:

```
<div id="title">My Company</div>
```

Customizing the Images

To replace the header image, open the `err.jsp` file and locate the following text in the body of the file.

```
<div><img src="/nesp/images/AccessMan_Login_Head.png"></div>
```

Replace the value of the `src` attribute with the path and filename of the image you want to use.

To replace the Novell logo image, locate the following text in the body of the file.

```
<div id="logo"><img src="/nesp/images/AccessMan31_Nlogo.png"></div>
```

Replace the value of the `src` attribute with the path and filename of the image you want to use.

Customizing the Colors

To change the background colors on the page, modify the color values in the `<style>` section of the `<head>`.

2.3.3 Customizing Tooltip Text for Authentication Contracts

The strings that users see when they mouse over the cards for authentication contracts can be customized. If you need to support only one language, modify the text in the Administration Console.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts*.
2. Click the name of a contract, then click *Authentication Card*.
3. Replace the English text in the *Text* option with the required language, then click *OK*.
4. Repeat Step 2 and Step 3 for each contract in the list.
5. Click *OK*, then update the Identity Server.

If you need to support multiple languages, you need to localize the tooltips. The nidsCardText attribute of the nidsAuthLocalContract object needs to be changed to a resource ID. The following procedure explains how to do this in the Administration Console. You can also use an LDAP browser.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts*.
2. Click the name of a contract, then click *Authentication Card*.
3. Replace the text in the *Text* option with a resource ID.

   For example, replace `Name/Password - Form` with `CUSTOM_NamePwdFormToolTip`.
4. Click *OK*.
5. Repeat Step 2 through Step 4 for each contract in the list.
6. Click *OK*, then update the Identity Server.
7. Use custom string resource files to define the localized strings:
   - **7a** Change to the `WEB-INF/classes` directory.
   - **7b** For each supported language, create a properties file. For example:

     ```
     nidp_custom_resources_fr.properties
     nidp_custom_resources_es.properties
     ```

     If you have already created these files for custom messages (see Section 2.3.1, “Customizing Messages,” on page 80), use the existing files.
   - **7c** For each resource ID you have created, add an entry that contains the resource ID and the text you want displayed for that language. For example:

     ```
     CUSTOM_NamePwdFormToolTip=Forma de Nombre/Clave
     ```
   - **7d** Repeat Step 7c for each supported language file.
8. Restart Tomcat.
   - **Linux Identity Server:** Enter the following command:

     ```
     /etc/init.d/novell-tomcat5 restart
     ```

- **Windows Identity Server:** Enter the following commands:

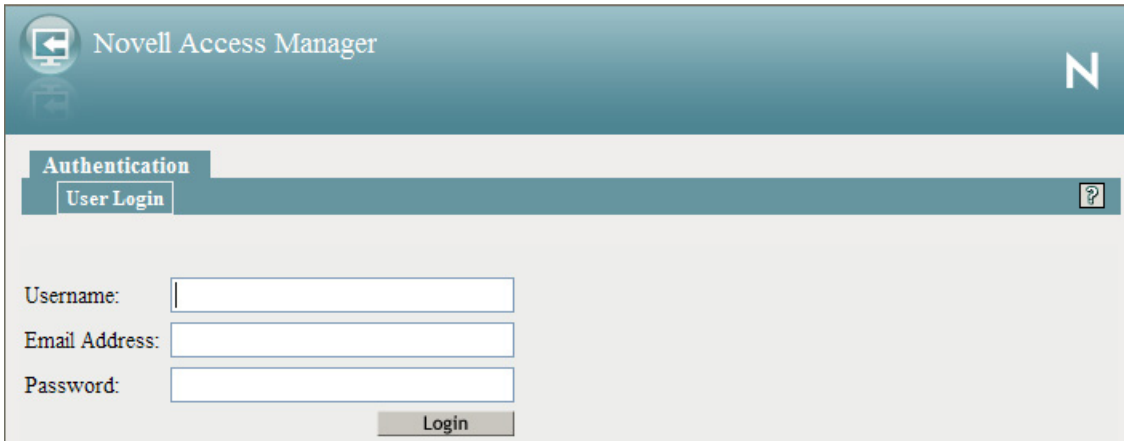
```
net stop Tomcat5
net start Tomcat5
```

2.4 Sample Custom Login Pages

2.4.1 Modified login.jsp File for Credential Prompts

The following code is a modified version of the 3.1 `login.jsp` file. It has been modified to add a prompt for the user’s email address. Figure 2-4 illustrates the login page that these changes produce.

Figure 2-4 *Custom Credentials*



Such a JSP file must be used with a contract that uses a method that defines the query for the new attribute. The method also needs to define which login file has been modified to display the prompt. For more information about this process, see “Customizing the Default Login Page to Prompt for Different Credentials” on page 61.

The sample code contains the following the text for the prompt:

```
<td align=left>
   <label>Email Address:</label>
</td>
```

It also adds an input element for the query variable:

```
<td align=left>
   <input type="text" class="smalltext" name="Ecom_User_Mail" size="30">
</td>
```

These elements are both part of the new `<tr>` element that has been added to the file. These lines are marked in bold in the following sample file.

```
<%@ page language="java" %>
<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>
<%@ page import="java.util.*" %>
<%@ page import="com.novell.nidp.*" %>
<%@ page import="com.novell.nidp.servlets.*" %>
<%@ page import="com.novell.nidp.resource.*" %>
<%@ page import="com.novell.nidp.resource.jsp.*" %>
<%@ page import="com.novell.nidp.ui.*" %>
<%
    ContentHandler handler = new ContentHandler(request,response);
    String target = (String) request.getAttribute("target");
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
<%=handler.getLanguageCode()%>">
<html lang="<%=handler.getLanguageCode()%>">
  <head>
        <META HTTP-EQUIV="Content-Language"
CONTENT="<%=handler.getLanguageCode()%>">
      <meta http-equiv="content-type" content="text/html;charset=utf-8">

    <style type="text/css" media="screen">
      td label       { font-size: 0.85em ; padding-right: 0.2em; }
      label     { font-size: 0.77em; padding-right: 0.2em; }
         input { font-family: sans-serif; }
      .instructions   { color: #4d6d8b; font-size: 0.8em; margin: 0 10px 10px
0 }
    </style>

       <script type="text/javascript" src="<%=
handler.getImage("showhide_2.js",false)%>"></script>
      <script language="JavaScript">
      var i = 0;
      function imageSubmit()
      {
            if (i == 0)
            {
              i = 1;
              document.IDPLogin.submit();
            }

            return false;
      }
    </script>
      </head>
  <body style="background-color: <%=handler.getBGColor()%>" marginwidth="0"
marginheight="0" leftmargin="0" topmargin="0" rightmargin="0"
onLoad="document.IDPLogin.Ecom_User_ID.focus();" >
        <form name="IDPLogin" enctype="application/x-www-form-urlencoded"
method="POST" action="<%= (String) request.getAttribute("url") %>"
AUTOCOMPLETE="off">
      <input type="hidden" name="option" value="credential">
<% if (target != null) { %>
      <input type="hidden" name="target" value="<%=target%>">
<% } %>
         <table border=0 style="margin-top: 1em" width="100%" cellspacing="0"
cellpadding="0">
            <tr>
            <td style="padding: 0px">
```

```
            <table border=0>
              <tr>
                <td align=left>
                  <label><%=handler.getResource(JSPResDesc.USERNAME)%></label>
                </td>
                <td align=left>
                  <input type="text" class="smalltext" name="Ecom_User_ID"
size="30">
                </td>
              </tr>
              <tr>
                <td align=left>
                  <label>Email Address:</label>
                </td>
                <td align=left>
                  <input type="text" class="smalltext" name="Ecom_User_Mail"
size="30">
                </td>
              </tr>
               <tr>
                <td align=left>
           <label><%=handler.getResource(JSPResDesc.PASSWORD)%></label>
                </td>
                <td align=left>
                <input type="password" class="smalltext" name="Ecom_Password"
size="30">
                </td>
              </tr>
              <tr>
                <td align=right colspan=2 style="white-space: nowrap">
                  <input alt="<%=handler.getResource(JSPResDesc.LOGIN)%>"
border="0" name="loginButton2" src="<%=
handler.getImage("btnlogin.gif",true)%>" type="image" value="Login"
onClick="return imageSubmit()">
                </td>
              </tr>
            </table>
          </td>
        </tr>
<%
    String err = (String)
request.getAttribute(NIDPConstants.ATTR_LOGIN_ERROR);
    if (err != null)
    {
%>
            <td style="padding: 10px">
            <div class="instructions"><%=err%></div>
            </td>
                    </tr>
<%  } %>
<%
  if (NIDPCripple.isCripple())
  {
%>
            <tr>
                   <td width="100%"
align="center"><%=NIDPCripple.getCrippleAdvertisement(request.getLocale())%><
/td>
            </tr>
```

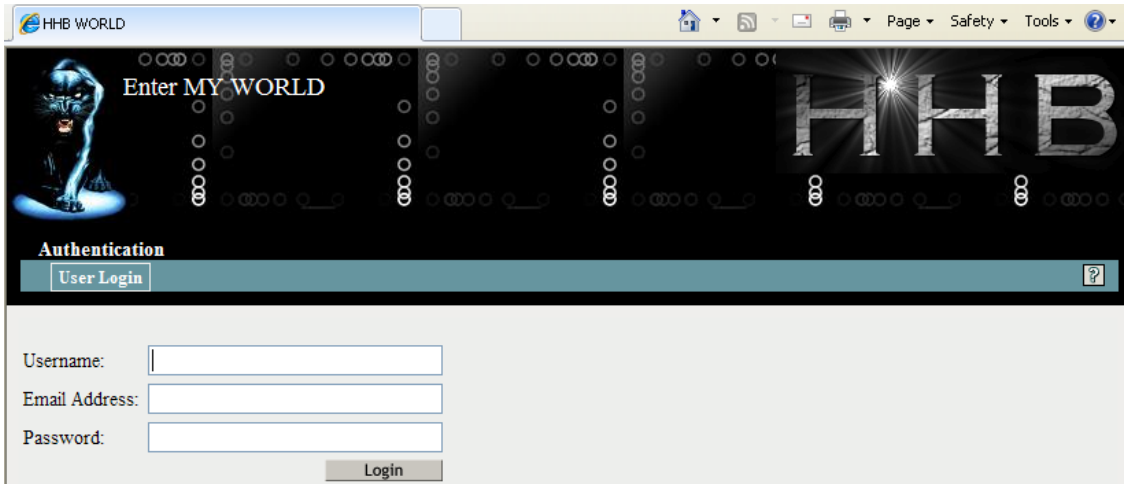
```
<%
  }
%>
          </table>
          </form>
      </body>
</html>
```

2.4.2 Custom nidp.jsp File with Custom Credentials

To create a custom `nidp.jsp` file that uses custom credentials, you need to modify the `nidp.jsp` file, create a method and contract for the file, and modify the `main.jsp` file. For instructions, see "Customizing the nidp.jsp File" on page 64 and "Adding Logic to the main.jsp File" on page 73.

Figure 2-5 illustrates the login page that the following custom `nidp.jsp` file and `main.jsp` file create.

Figure 2-5 *Custom Branding with Custom Credential Prompts*



The credential frame uses the same modifications in the sample from Section 2.4.1, "Modified login.jsp File for Credential Prompts," on page 85. The following sections provide the other required sample files to create this login page and information on the required method and contract:

- "The Modified nidp.jsp File" on page 88
- "The Modified main.jsp File" on page 94
- "The Method and the Contract" on page 95

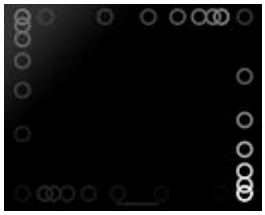
The Modified nidp.jsp File

The background, menu, and border colors are set to black. These colors are specified in the following lines in the sample file:

```
// Background color
String bgcolor    = "#000000";

// Menu color
String menucolor    = "#000000";

// Border color
String bcolor    = "#000000";
```

Figure 2-6 illustrates the image (`images2.jpeg`) that this custom page uses for the header background image:

Figure 2-6 *Background Image*



This image is the repeatable image that allows the header to be resized. This image is specified in the following lines in the file:

```
// The header background image that gets repeated
   String hdrBgndImg   = "/custom_images/images2.jpeg";
```

Figure 2-7 illustrates the image (`images3.jpeg`) that this custom page uses for the product logo that appears on left of the header frame.

Figure 2-7 *Header Image*

Figure 2-8 illustrates the image (`hhbimages.jpeg`) that this custom page uses to replace the Novell company logo on the right of the header frame.

Figure 2-8 *Company Logo*

The following lines define what appears as the title for the browser window:

```
<title>HHB WORLD</title>
```

The following line defines the header title value:

```
String hdrTitle     = "Enter MY WORLD";
```

Its position is controlled by the following line in the file:

```
 #title { position: absolute; font-size: 1.2em; color: white; top: 18px; left:
85px; }
```

The top position has been modified from 13px to 18px and the left position has been modified from 55px to 85px. The other lines in this section control the position of the other items in the header.

The lines that have been modified are marked in bold in the following file.

```
<%
    ContentHandler handler = new ContentHandler(request,response);

    // Background color
    String bgcolor   = "#000000";

    // Menu color
    String menucolor   = "#000000";

    // Border color
    String bcolor   = "#000000";

    // The header background image that gets repeated
    String hdrBgndImg  = "/custom_images/images2.jpeg";

    String hdrImage  = "/custom_images/images3.jpeg";

    String hdrLogo      = "/custom_images/hhbimages.jpeg";

    String hdrTitle     = "Enter MY WORLD";

    String query     =  request.getQueryString();
    if (query != null && query.length() > 0)
        query = "&" + query;
    else query = "";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//Dtd HTML 4.0 transitional//
<%=handler.getLanguageCode()%>">
<html lang="<%=handler.getLanguageCode()%>">
    <head>
        <title>HHB WORLD</title>
        <meta http-equiv="content-type" content="text/html; charset=UTF-8">
        <link href="<%= handler.getImage("hf_menu.css",false)%>"
rel="stylesheet">
        <link href="<%= handler.getImage("HF_message.css",false)%>"
rel="stylesheet">
        <link href="<%= handler.getImage("HF_obj_list_table.css",false)%>"
rel="stylesheet">
    <style>
    * { margin: 0; padding: 0; }
        #header    { background-image: url(<%=
handler.getImage(hdrBgndImg,false)%>); background-repeat: repeat-x; }
        #logo      { position: absolute; top: 0px; right: 0px;  }
        #title         { position: absolute; font-size: 1.2em; color: white;
top: 18px; left: 85px; }
    #subtitle    { position: relative; font-size: .9em; color: black; white-
space: nowrap; top: 0px; left: 0px; text-align: right; }
    #mcontent     { position: relative; padding: 5px; background-color:
```

```
<%=bgcolor%>; }
    #content      { width: 100%; border: 0; margin: 0; padding: 0; overflow:
none; height: 376px; background-color: <%=bgcolor%>;}
    #logoutbut    { position: absolute; top: 25px; right: 35px;  }
   #helpbutlogin  { position: absolute; color: yellow; top: 25px; right: 10px;
}
    #loggingbut    { position: absolute; color: blue; top: 25px; right: 65px;
}

    .NLtab .tab1s    { background-color: <%=menucolor%>; padding-left: 3px;
padding-right: 8px; text-align: center; white-space: nowrap;   }
    .NLtab .tab1s a   { text-decoration: none; }
    .NLtab span.tab1s { padding:5; color: white; font-size: 0.9em; font-
weight: bold; line-height: 17px; background-color: transparent; background-
image: none; text-decoration: none; }
    .NLtab .tab1u   { background-color: <%=bgcolor%>; padding-left: 3px;
padding-right: 3px; text-align: center; white-space: nowrap; border-left: 1px
solid <%=bcolor%>; border-right: 1px solid <%=bcolor%>; border-top: 1px solid
<%=bcolor%>;          }
    .NLtab span.tab1u { border: none; padding:5; color: black; font-size:
0.8em; font-weight: bold; line-height: 17px; text-decoration: none;
background-color: transparent;  }

    .NLtab tr.subtab td  {  color: white; padding: 2px }
   .NLtab tr.subtab a  { font-size: .8em; color: white; text-decoration: none;
padding: 2px 5px 2px 5px}

   .selx  { border: 1px solid rgb(239, 238, 236); font-size: 1em; font-weight:
bolder; background-repeat: repeat-x; background-position: 0pt bottom;}
    .unselx  { border: 0px; font-size: .9em; font-weight: normal; background-
image: none; }
        </style>

        <script>
         var g_curCard = null;     // initial displayed card
     var g_cardContainer = null;   // div that holds all the authentication
cards
     var g_curSubtab = null;   // subtab currently displayed
     var g_curTab = null;     // tab currently displayed

         var menuItem = 0;
             function showHide(i)
             {
                 document.getElementById('menu1').style.display='none';
                 document.getElementById('menu2').style.display='none';
                 document.getElementById('submenu1').style.display='none';
                 document.getElementById('submenu2').style.display='none';
                 document.getElementById('menu' + i).style.display='block';
                document.getElementById('submenu' + i).style.display='block';
         if (i == 1)
             switchContentPage("<%= handler.getJSP("content")%>");
         else
             switchContentPage("<%= handler.getJSP("IdentityEditor")%>");
             }

     function switchContentPage(newSrc)
     {
           parent.document.getElementById("content").src = newSrc;
     }
```

```
function onloadhandler()
{
    g_cardContainer = document.getElementById("cardcontainer");
    g_curSubtab     = document.getElementById("loginsubtab");
    g_curTab        = document.getElementById("authtab");
    g_curCard       = document.getElementById("selectedCard0");
}

 function showhideTab(divid)
{
    var element1 = document.getElementById(divid);

    if(element1.style.display == "none")
    {
        element1.style.display = "block";
    g_curTab.style.display = "none";

        g_curTab = element1;
    }
}

function subtabchange(divid)
 {
    var element1 = document.getElementById(divid);
    var element2 = g_curSubtab;
    element1.className = "selx";
    if (element1.id != element2.id)
    {
        element2.className = "unselx";
    }
    g_curSubtab = element1;
}

function showHelp()
 {
    var helpURL = "login.html";
            if (g_curSubtab.id == "fedsubtab")
        helpURL = "<%=handler.getHelp("federations.html")%>";

            else if (g_curSubtab.id == "myprofile")
    helpURL = "<%=handler.getHelp("myprofile.html")%>";

            else if (g_curSubtab.id == "sharing")
     helpURL = "<%=handler.getHelp("sharing.html")%>";

            else if (g_curSubtab.id == "loginsubtab")
    helpURL = "<%=handler.getHelp("userlogin.html")%>";

            else if (g_curSubtab.id == "newcardsubtab")
    helpURL = "<%=handler.getHelp("newcard.html")%>";

          else if (g_curSubtab.id == "logTicketsubtab")
        helpURL = "<%=handler.getHelp("logticket.html")%>";

        var w;
        w = window.open(helpURL, "nidsPopupHelp",
"toolbar=no,location=no,directories=no,menubar=no,scrollbars=yes,resizable=ye
s,width=500,height=500");
```

```
            if (w != null)
            {
             w.focus();
            }
     }
        </script>
    </head>

    <body onload="onloadhandler()">
        <table width=100% border=0 cellpadding=0 cellspacing=0
bgcolor=<%=bgcolor%> >
            <tr>
        <td>
            <table cellspacing=0 width=100% border=0>
            <tr>
            <td width=100%>
                <div id="header"><img
src="<%=handler.getImage(hdrImage,false)%>"></div>
                <div id="logo"><img
src="<%=handler.getImage(hdrLogo,false)%>"></div>
                <div id="title"><%=hdrTitle%></div>
            </td>
        </tr>
        </table>
        </td>
    </tr>
    <tr>
        <td>
            <table cellspacing=5 width=100%>
            <tr>
            <td>
                <%@ include file="menus.jsp" %>
            </td>
        </tr>
        </table>
        </td>
    </tr>
    <tr>
        <td>
            <table cellspacing=0 border=0 width=100%>
            <tr>
            <td>
                        <iframe scrolling=no id="content"
src="<%=handler.addCardParm(handler.getJSP(handler.isJSPMsg() ?
handler.getJSPMessage().getJSP() : NIDPConstants.JSP_CONTENT)) + query%>"
frameborder=0></iframe>
            </td>
        </tr>
        </table>
        </td>
    </tr>
    </table>
    </body>
</html>
```

The Modified main.jsp File

The following sample file has two types of modifications. The following line has been added so that the URI of the contract can be read and used as a condition for selecting the login page to display:

```
String strContractURI = hand.getContractURI();
```

The following lines define the login page to use when the URI of the contract is set to login/custom.

```
else if(strContractURI != null && strContractURI.equals("login/custom"))
    {
%>
     <%@ include file="custom.jsp" %>

<%  }
```

The lines that have been added are marked in bold in the following file.

```
<%@ page language="java" %>
<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>
<%@ page import="com.novell.nidp.*" %>
<%@ page import="com.novell.nidp.resource.jsp.*" %>
<%@ page import="com.novell.nidp.ui.*" %>
<%@ page import="com.novell.nidp.common.util.*" %>
<%@ page import="com.novell.nidp.liberty.wsf.idsis.apservice.schema.*" %>

<%
    ContentHandler hand = new ContentHandler(request,response);
    String strContractURI = hand.getContractURI();

    // Is there a JSP defined on a class definition or a method definition
    // that should be displayed as the main jsp here?
    if (hand.contractDefinesMainJSP())
    {
%>

        <%@ include file="mainRedirect.jsp" %>
<%  }

else if(strContractURI != null && strContractURI.equals("login/custom"))
    {
%>
     <%@ include file="custom.jsp" %>

<%  }

    // This is the jsp used by default
    else
    {
%>
        <%@ include file="nidp.jsp" %>
<%  }  %>
```

The Method and the Contract

After modifying the two files, you still need to create a method and a contract. The method needs to use a name/password class and have the following properties defined:

- Query property values:

  Property Name: `Query`

  Property Value: `(&(objectclass=person)(mail=%Ecom_User_Mail%))`

- JSP property values:

  Property Name: `JSP`

  Property Value: <*filename*>

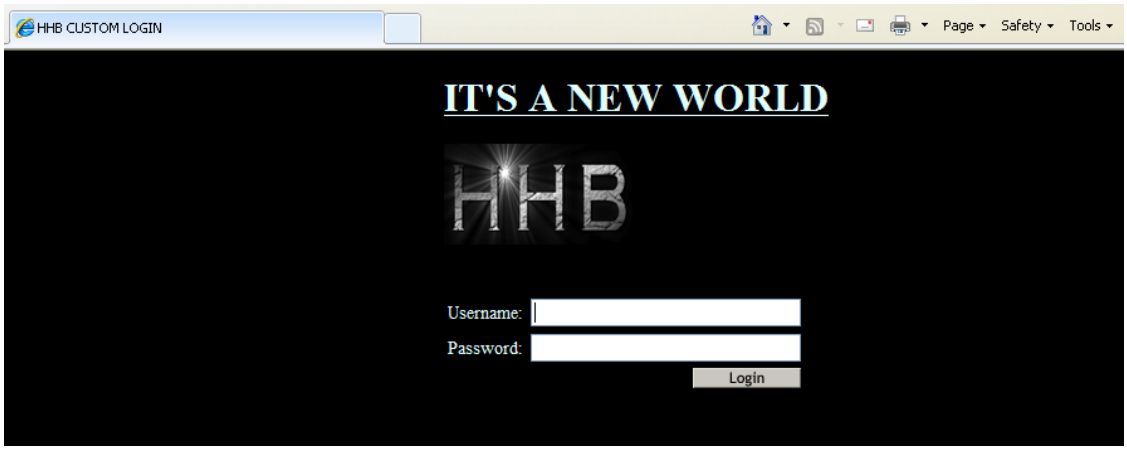
  Replace <*filename*> with the name of your login page that modifies the credential prompts. Do not include the JSP extension in the value.

You then need to create a contract that uses this method and assign it to a protected resource.

2.4.3 Custom 3.1 login.jsp File

To create this type of page, you need to start with the `login.jsp` file that ships with Access Manager 3.1 and then add the required code for a header. Figure 2-9 illustrates such a page.

Figure 2-9 *Custom Page Derived from the 3.1 login.jsp File*



To create this page, see the following sections:

- "The Modified login.jsp File" on page 95
- "The Method and the Contract" on page 98

The Modified login.jsp File

This custom page does not modify the credential frame. The lines that define the window title (HHB CUSTOM LOGIN), the page header title (IT'S A NEW WORLD), and the image (`hhbimages.png`) are marked in bold in the following sample file.

```
<%@ page language="java" %>
<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>
<%@ page import="java.util.*" %>
<%@ page import="com.novell.nidp.*" %>
<%@ page import="com.novell.nidp.servlets.*" %>
<%@ page import="com.novell.nidp.resource.*" %>
<%@ page import="com.novell.nidp.resource.jsp.*" %>
<%@ page import="com.novell.nidp.ui.*" %>
<%
    ContentHandler handler = new ContentHandler(request,response);
    String target = (String)request.getAttribute("target");
    String hdrImage = "/custom_images/hhbimages.jpeg";

 %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
<%=handler.getLanguageCode()%>">
<html lang="<%=handler.getLanguageCode()%>">
   <head>
           <title>HHB CUSTOM LOGIN </title>
      <META HTTP-EQUIV="Content-Language"
CONTENT="<%=handler.getLanguageCode()%>">
      <meta http-equiv="content-type" content="text/html;charset=utf-8">



      <style type="text/css" media="screen">


         td label       { font-size: 0.85em ; padding-right: 0.2em; }
         label       { font-size: 0.77em; padding-right: 0.2em; }
            input { font-family: sans-serif; }
        .instructions  { color: #4d6d8b; font-size: 0.8em; margin: 0 10px 10px
0 }
      </style>

      <script type="text/javascript" src="<%=
handler.getImage("showhide_2.js",false)%>"></script>
      <script language="JavaScript">
         var i = 0;
         function imageSubmit()
         {
             if (i == 0)
             {
                i = 1;
                document.IDPLogin.submit();
             }

             return false;
         }
      </script>
      </head>
   <body text="lightcyan" style="background-color:Black" marginwidth="300"
marginheight="100" leftmargin="350" topmargin="0" rightmargin="0"
onLoad="document.IDPLogin.Ecom_User_ID.focus();" >
   <br>
         <h1><u>       IT'S A NEW WORLD</u></h1>

       <form name="IDPLogin" enctype="application/x-www-form-urlencoded"
```

```
method="POST" action="<%= (String) request.getAttribute("url") %>"
AUTOCOMPLETE="off">
         <input type="hidden" name="option" value="credential">
<% if (target != null) { %>
         <input type="hidden" name="target" value="<%=target%>">
<% } %>
           <table border=0 style="margin-top: 1em" width="20" cellspacing="0"
cellpadding="0">
                                 <tr>
                                     <div id="headimage"><img
src="<%=handler.getImage(hdrImage,false)%>" alt="" height="80" width="150"
border="0"></div>
                                                 </tr>
                                           <tr>

                                          <td style="padding: 0px">
                  <table border=0>
                  <br><br>

                     <tr>
                        <td align=center>

<label><%=handler.getResource(JSPResDesc.USERNAME)%></label>
                        </td>
                        <td align=center>
                           <input type="text" class="smalltext"
name="Ecom_User_ID" size="30">
                        </td>
                     </tr>
                     <tr>
                        <td align=center>

<label><%=handler.getResource(JSPResDesc.PASSWORD)%></label>
                        </td>
                        <td align=center>
                           <input type="password" class="smalltext"
name="Ecom_Password" size="30">
                        </td>
                     </tr>
                     <tr>
                        <td align=right colspan=2 style="white-space: nowrap">
                           <input
alt="<%=handler.getResource(JSPResDesc.LOGIN)%>" border="0"
name="loginButton2" src="<%= handler.getImage("btnlogin.gif",true)%>"
type="image" value="Login" onClick="return imageSubmit()">
                        </td>
                     </tr>
                  </table>
               </td>
            </tr>
<%
    String err = (String)
request.getAttribute(NIDPConstants.ATTR_LOGIN_ERROR);
    if (err != null)
    {
%>
               <td style="padding: 10px">
                  <div class="instructions"><%=err%></div>
               </td>
```

```
                        </tr>
<%  } %>
<%
   if (NIDPCripple.isCripple())
   {
%>
                <tr>
                        <td width="100%"
align="center"><%=NIDPCripple.getCrippleAdvertisement(request.getLocale())%><
/td>
                </tr>
<%
   }
%>
            </table>
         </form>
      </body>
</html>
```

The Method and the Contract

After modifying the file, you still need to create a method and a contract. The method needs to use a name/password class and have the following properties defined:

- JSP property values:

  Property Name: `JSP`

  Property Value: <*filename*>

  Replace <*filename*> with the name of your custom login page. Do not include the JSP extension in the value.

- MainJSP property values:

  Property Name: `MainJSP`

  Property Value: `true`

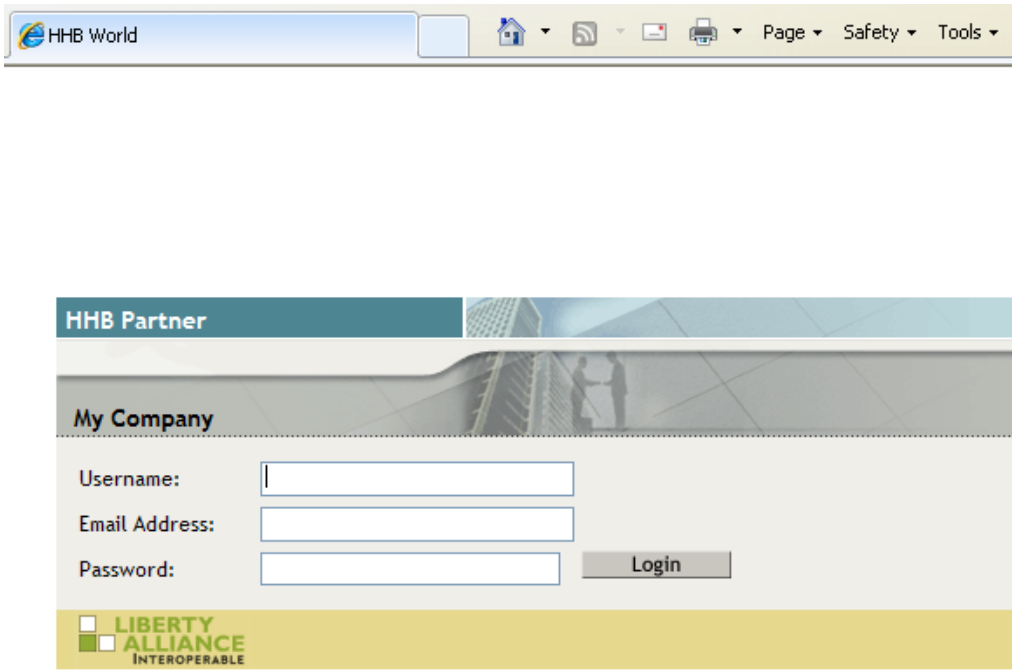
You then need to create a contract that uses this method and assign it to a protected resource.

2.4.4 Custom 3.0 login.jsp File

To create this type of page, you need to start with the `login.jsp` file that shipped with Access Manager 3.0. This file needs to be modified to run on Access Manager 3.1.*x*. For instructions, see "Modifications Required for a 3.0 Login Page" in the *Novell Access Manager 3.1 SP2 Installation Guide*.

Figure 2-10 illustrates a page that has been modified to remove the Novell branding and logo. It has also been modified to prompt the user for an e-mail address in addition to a username and password.

Figure 2-10 *Custom Page Derived from the 3.0 login.jsp File*



To create this page, see the following sections:

- "Modifying the File" on page 99
- "The Method and the Contract" on page 102

Modifying the File

The bold lines in the following sample file are the lines that have been modified to change the branding and the login prompts.

```
<%@ page language="java" %>
<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>
<%@ page import="com.novell.nidp.common.provider.*" %>
<%@ page import="java.util.*" %>
<%@ page import="com.novell.nidp.ui.*" %>
<%@ page import="com.novell.nidp.*" %>
<%@ page import="com.novell.nidp.servlets.*" %>
<%@ page import="com.novell.nidp.resource.*" %>
<%@ page import="com.novell.nidp.resource.jsp.*" %>
<%@ page import="com.novell.nidp.common.xml.w3c.*" %>
<%
    ContentHandler handler = new ContentHandler(request,response);
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
<%=handler.getLanguageCode()%>">
<html lang="<%=handler.getLanguageCode()%>">
    <head>
    <link rel="stylesheet" href="<%= request.getContextPath() %>/images/
hf_style.css" type="text/css">
    <style type="text/css" media="screen"><!--
      #headimage    { position: relative; top: 0px; left: 0px; z-index: 1}
```

```
      #title    { position: relative; top: 40px; left: 5px; color: white; z-
index: 4}
      #locallabel    { position: relative; top: 78px; left: 10px; z-index: 4}
      #login       { text-align: center }
         --></style>
    <META HTTP-EQUIV="Content-Language"
CONTENT="<%=handler.getLanguageCode()%>">
    <title>MY WORLD</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8">
    <script type="text/javascript" src="<%= request.getContextPath() %>/
images/showhide_2.js"></script>
    <script language="JavaScript">

      var i = 0;
      function imageSubmit()
      {
          if (i == 0)
          {
             i = 1;
             document.IDPLogin.submit();
          }

          return false;
      }
  </script>
    </head>
    <body marginwidth="0" marginheight="0" leftmargin="0" topmargin="0"
rightmargin="0" onLoad="document.IDPLogin.Ecom_User_ID.focus();" >
        <form name="IDPLogin" enctype="application/x-www-form-urlencoded"
method="POST" action="<%= (String) request.getAttribute("url") %>"
AUTOCOMPLETE="off">
       <table style="margin-top: 6em" width="100%" border="0" cellspacing="0"
cellpadding="0">
        <tr>
            <td width="50%" height="80 px"> </td>
        <td colspan="2">
            <div id="title"><b>HHB Partner</b></div>
        <div id="locallabel"><b>My Company</b></div>
        <div id="headimage"><img src="<%= request.getContextPath() %>/images/
Odyssey_Head.gif" alt="" height="80" width="550" border="0"></div>
                    </td>
        <td width="100%"> </td>
        </tr>
        <tr>
            <td width="50%"> </td>
        <td style="background-color: #efeee9; padding: 10px" colspan="2">
<%
    String err = (String)
request.getAttribute(NIDPConstants.ATTR_LOGIN_ERROR);
    if (err != null)
    {
%>
        <div><label><%=err%></label></div>
<%  }

    // Determine if this login page is being used for account identification
    // purposes
 %>
          <span id="login2" style="display: block;">
```

```
            <table>
              <tr>
                <td nowrap="nowrap">
                  <div>
                    <label style="width:
100px"><%=handler.getResource(JSPResDesc.USERNAME)%></label></label>
                  </div>
                </td>
                <td width="100%" nowrap="nowrap">
                  <div>
                    <input type="text" class="smalltext" name="Ecom_User_ID"
size="30">
                  </div>
                </td>
              </tr>
              <tr>
                <td nowrap="nowrap">
                  <div>
                    <label style="width: 100px">Email Address:</label></label>
                  </div>
                </td>
                <td width="100%" nowrap="nowrap">
                  <div>
                   <input type="text" class="smalltext" name="Ecom_User_Mail"
size="30">
                  </div>
                </td>
              </tr>

              <tr>
                <td nowrap="nowrap">
                  <div>
                    <label><%=handler.getResource(JSPResDesc.PASSWORD)%></
label>
                  </div>
                </td>
                <td style="white-space: nowrap">
                  <div>
                    <input type="password" class="smalltext"
name="Ecom_Password" size="30">  
                    <input alt="<%=handler.getResource(JSPResDesc.LOGIN)%>"
border="0" name="loginButton2"
src="<%=handler.getImage("btnlogin.gif",true)%>"
 type="image" value="Login" onClick="return imageSubmit()">
                  </div>
                </td>
              </tr>
            </table>
          </span>
        </td>
        <td width="100%"> </td>
      </tr>

        <tr>
            <td width="50%"></td>
        <td style="background-color: #E6D88C; padding-left: 10px"><img
style="padding-right: 200px" src="<%= request.getContextPath() %>/images/
LAP_interoperable_logo_100.gif" align="absmiddle" border="0"></td>
        <td style="background-color: #E6D88C; padding-right: 10px"
```

```
align="right" width="100">


        </td>
            <td width="100%"></td>
        </tr>
<%
  if (NIDPCripple.isCripple())
  {
%>
        <tr>
               <td colspan=4 width="100%"
align="center"><%=NIDPCripple.getCrippleAdvertisement(request.getLocale())%><
/td>
        </tr>
<%
  }
%>
        </table>
        </form>
    </body>
</html>
```

The Method and the Contract

After modifying the file, you still need to create a method and a contract. The method needs to use a name/password class and have the following properties defined:

- Query property values:

  Property Name: `Query`

  Property Value: `(&(objectclass=person)(mail=%Ecom_User_Mail%))`

- JSP property values:

  Property Name: `JSP`

  Property Value: <*filename*>

  Replace <*filename*> with the name of your custom login page. Do not include the JSP extension in the value.

- MainJSP property values:

  Property Name: `MainJSP`

  Property Value: `true`

You then need to create a contract that uses this method and assign it to a protected resource.

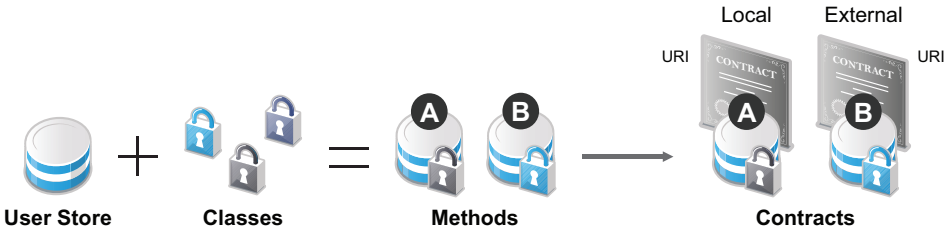
Configuring Local Authentication

3

To guard against unauthorized access, Access Manager supports a number of ways for users to authenticate. These include name/password, RADIUS token-based authentication, and X.509 digital certificates. You configure authentication at the Identity Server by creating authentication contracts that the components of Access Manager (such as an Access Gateway) can use to protect a resource.

Figure 3-1 illustrates the components of a contract:

Figure 3-1 *Local Authentication*



- **User stores:** The user directories to which users authenticate on the back end. You set up your user store when you create an Identity Server cluster configuration. See Section 3.1, "Configuring Identity User Stores," on page 104.
- **Classes:** The code (a Java class) that implements a particular authentication type (name/password, RADIUS, and X.509) or means of obtaining credentials. Classes specify how the Identity Server requests authentication information, and what it should do to validate those credentials. See Section 3.2, "Creating Authentication Classes," on page 118.
- **Methods:** The pairing of an authentication class with one or more user stores, and whether the method identifies a user. See Section 3.3, "Configuring Authentication Methods," on page 123.
- **Contracts:** The basic unit of authentication. Contracts can be local (executed at the server) or external (satisfied by another Identity Server). Contracts are identified by a unique URI that can be used by Access Gateways and agents to protect resources. Contracts are comprised of one or more authentication methods used to uniquely identify a user. You can associate multiple methods with one contract. See Section 3.4, "Configuring Authentication Contracts," on page 125.

This section also explains the following:

- "Using a Password Expiration Service" on page 128
- "Using Activity Realms" on page 130
- "Specifying Authentication Defaults" on page 131
- "Managing Direct Access to the Identity Server" on page 134

3.1 Configuring Identity User Stores

User stores are LDAP directory servers to which end users authenticate. You must specify an initial user store when creating an Identity Server configuration. You use the same procedure for setting up the initial user store, adding a user store, or modifying an existing user store.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Servers* > *Edit* > *Local*.
2. Select from the following actions:

   New: To add a user store, click *New*. For configuration information, see Section 3.1.2, "Configuring the User Store," on page 105.

   Delete: To delete a user store, select the user store, then click *Delete*.The user store list needs to contain at least one configured user store for the Identity Server to be functional.

   Modify: To modify the configuration of an existing user store, click the name of a user store. For configuration information, see Section 3.1.2, "Configuring the User Store," on page 105.
3. Click *OK*, then update the Identity Server if you have modified the configuration.

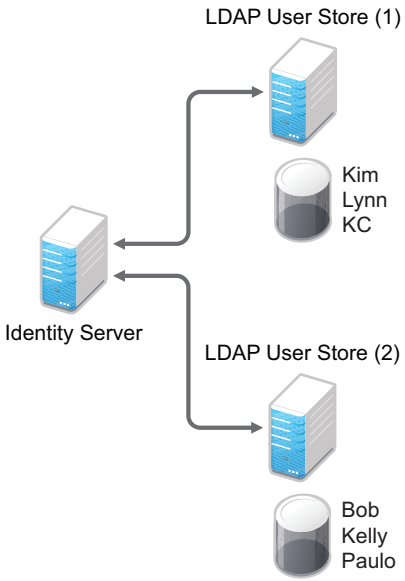
See the following sections for specific configuration tasks:

- Section 3.1.1, "Using More Than One LDAP User Store," on page 104
- Section 3.1.2, "Configuring the User Store," on page 105
- Section 3.1.3, "Configuring an Admin User for the User Store," on page 109
- Section 3.1.4, "Configuring a User Store for Secrets," on page 109

3.1.1 Using More Than One LDAP User Store

You can configure the Identity Server to search more than one user store during authentication. Figure 3-2 illustrates this type of configuration.

Figure 3-2 *Multiple LDAP Directories*



It is assumed that each LDAP directory contains different users. You should make sure the users have unique names across all LDAP directories. If both directories contain a user with an identical name, the name and password information discovered in the search of the first directory is always used for authentication. You specify the search order when configuring the authentication method.

When users are added to the configuration store, objects are created for Access Manager profiles. If you delete a user from the LDAP directory, orphaned objects for that user remain in the configuration store. Ensure that you also delete those objects from the configuration store. See "Orphaned Objects in the Trust/Configuration Store" in the *Novell Access Manager 3.1 SP2 Administration Console Guide*.

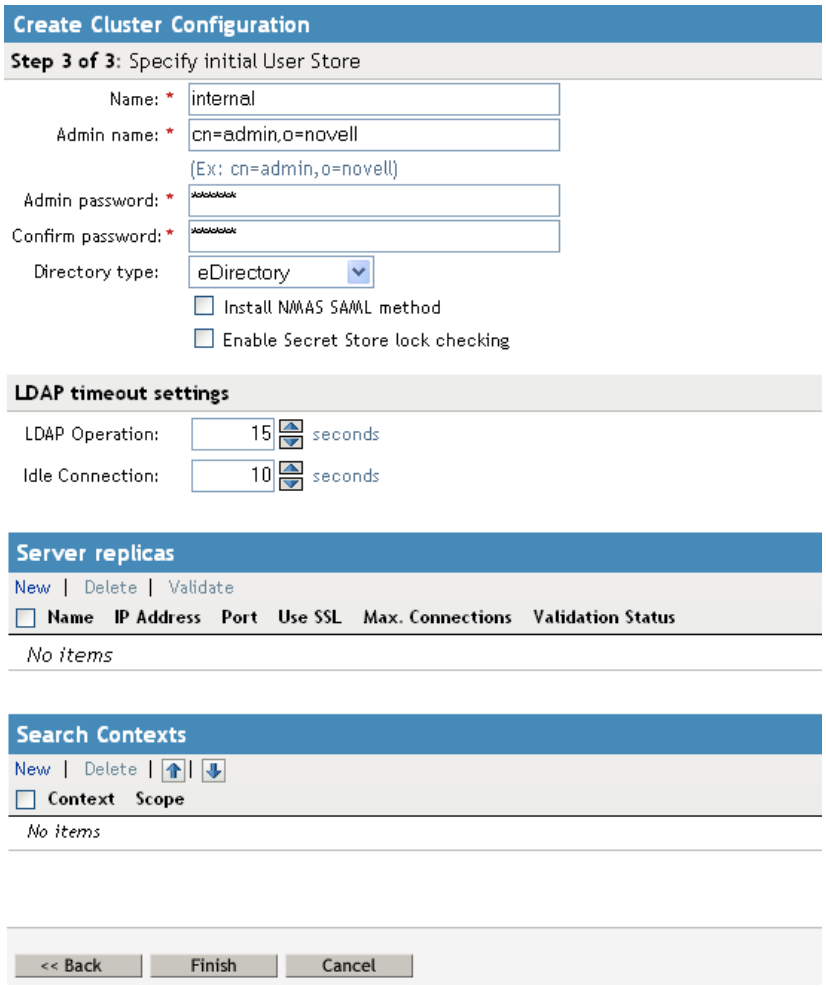
If you add a secondary Administration Console and you have added replicas to the user store of the primary Administration Console, ensure that you also add the replicas to the secondary Administration Console.

All user stores that you add are included in health checks. If health problems are found, the system displays the user store on the Health page and in the trace log file.

3.1.2 Configuring the User Store

1. In the Administration Console, click *Devices* > *Identity Servers* > *Servers* > *Edit* > *Local*.
2. In the *User Stores* list, click *New* or the name of an existing user store.

   If you are creating an Identity Server configuration, this is Step 3 of the wizard.



3 Fill in the following fields:

Name: The name of the user store for reference.

Admin Name: The distinguished name of the admin user of the LDAP directory, or a proxy user with specific LDAP rights to perform searches. Administrator-level rights are required for setting up a user store. This ensures read/write access to all objects used by Access Manager. For more information about this user, see Section 3.1.3, “Configuring an Admin User for the User Store,” on page 109.

Each directory type uses a slightly different format for the DN:

- **eDirectory:** cn=admin,ou=users,o=novell
- **Active Directory:** cn=Administrator,cn=users,dc=domeh,dc=test,dc=com

  or cn=john smith,cn=users,dc=domeh,dc=test,dc=com
- **Sun ONE:** cn=admin,cn=users,dc=novell,dc=com

Admin Password and Confirm Password: Specify the password for the admin user and confirm it.

Directory Type: The type of LDAP directory. You can select *eDirectory*, *Active Directory*, or *Sun ONE*. If you have installed an LDAP server plug-in, you can select the custom type that you have configured it to use. For more information, see LDAP Server Plug-In (http://developer.novell.com/documentation/nacm31/nacm_enu/data/bfg38fg.html).

If eDirectory has been configured to use Domain Services for Windows, eDirectory behaves like Active Directory. When you configure such a directory to be a user store, its *Directory Type* must be set to Active Directory for proper operation.

Install NMAS SAML method: (eDirectory only) Extends the schema on the eDirectory server and installs an NMAS method. This method converts the Identity Server credentials to a form understood by eDirectory. This method is required if you have installed Novell SecretStore on the eDirectory server and you are going to use that SecretStore for Access Manager secrets. If you select this option, make sure the admin you have configured for the user store has sufficient rights to extend the schema and add objects to the tree.

For additional configuration steps required to use secrets, see Section 3.1.4, "Configuring a User Store for Secrets," on page 109.

Enable Secret Store lock checking: (eDirectory only) Enables Access Manager to prompt users for a passphrase when secrets are locked.

- If Access Manager is sharing secrets with other applications and these applications are using the security flag that locks secrets when a user's password is reset, you need to enable this option.
- If Access Manager is not sharing secrets with other applications, the secrets it is using are never locked, and you do not need enable this option.

4 Under *LDAP timeout settings*, specify the following:

LDAP Operation: Specify how long in seconds a transaction can take before timing out.

Idle Connection: Specify how long in seconds before connections begin closing. If a connection has been idle for this amount of time, the system creates another connection.

5 To specify a server replica, click *New*, then fill in the following fields:

For an eDirectory server, you should use a replica of the partition where the users reside. Ensure that each LDAP server in the cluster has a valid read/write replica. One option is to create a users partition (a partition that points to the OU containing the user accounts) and reference this server replica.

Name: The display name for the LDAP directory server. If your LDAP directory is replicated on multiple servers, use this name to identify a specific replica.

IP Address: The IP address of the LDAP directory server.

Port: The port of the LDAP directory server. Specify 389 for the clear text port, and 636 for the encrypted port.

Use secure LDAP connections: Specifies that the LDAP directory server requires secure (SSL) connections with the Identity Server.

This is the only configuration we recommend for the connection between the Identity Server and the LDAP server in a production environment. If you use port 389, usernames and passwords are sent in clear text on the wire.

This option must be enabled if you use this user store as a Novell SecretStore User Store Reference in the Credential Profile details. (See Section 13.3, "Configuring Credential Profile Security and Display Settings," on page 304.) If you have specified that this user store is a SecretStore User Store Reference, this option is enabled but not editable.

Connection limit: The maximum number of pooled simultaneous connections allowed to the replica. Valid values are between 5 and 100. How many you need depends upon the speed of your LDAP servers. If you modify the default value, monitor the change in performance. Larger numbers do not necessarily increase performance.

6 Click *Auto import trusted root*.

7 Click *OK* to confirm the import.

8 Select one of the certificates in the list.

You are prompted to choose either a server certificate or a root CA certificate. To trust one certificate, choose *Server Certificate*. Choose *Root CA Certificate* to trust any certificate signed by that certificate authority.

9 Specify an alias, then click *OK*.

10 Click *OK* in the *Specify server replica information* dialog box.

11 Select the replica, then click *Validate* to test the connection between the Identity Server and the replica.

The system displays the result under *Validation Status*. The system displays a green check mark if the connection is valid.

12 (Optional) To add additional replicas for the same user store, repeat Step 5 through Step 11.

Adding multiple replicas adds load balancing and failover to the user store. Replicas must be exact copies of each other.

For load balancing, a hash algorithm is used to map a user to a replica. All requests on behalf of that user are sent to that replica. Users are moved from their replica to another replica only when their replica is no longer available.

13 Add a search context.

The search context is used to locate users in the directory when a contract is executed.

- If a user exists outside of the specified search context (object, subtree, one level), the Identity Server cannot find the user, and the user cannot log in.
- If the search context is too broad, the Identity Server might find more than one match, in which case the contract fails, and the user cannot log in.

For example, if you allow users to have the same username and these users exist in the specified search context, these users cannot log in if you are using a simple username and password contract. The search for users matching this contract would return more than one match. In this case, you need to create a contract that specifies additional attributes so that the search returns only one match. For more information on how to create such contracts, see Section 15.3.1, “Authentication Classes and Duplicate Common Names,” on page 363.

IMPORTANT: For Active Directory, do not set the search context at the root level and set the scope to Subtree. This setting can cause serious performance problems. It is recommended that you set multiple search contexts, one for each top-level organizational unit.

---

14 Click *Finish*.

15 If prompted to restart Tomcat, click *OK*. Otherwise, update the Identity Server.

3.1.3 Configuring an Admin User for the User Store

The Identity Server must log in to each configured user store. It searches for users, and when a user is found, it reads the user's attributes values. When you configure a user store, you must supply the distinguished name of the user you want the Identity Server to use for logging in. You can use the admin user of your user store, or you can create a specialized admin user for the this purpose. When creating this admin user, you need to grant the following rights:

- The admin user needs rights to browse the tree, so the Identity Server can find the user who is trying to authenticate. The admin user needs browse rights to the object class that defines the users and read and compare rights to the attributes of that class. When looking for the user, the Identity Server uses the GUID and naming attributes of the user class.

Directory	Object Class	GUID Attribute	Naming Attribute
eDirectory	User	guid	cn
Active Directory	User	objectGUID	sAMAccountName
Sun ONE	inetOrgPerson	nsuniqueid	uid

- The admin user needs read rights to any attributes used in policies (Role, Form Fill, Identity Injection, Authorization).
- If a secret store is used in Form Fill policies, the admin user needs write rights to the attributes storing the secrets.
- If a password management servlet is enabled, the admin user needs read rights to the attributes controlling grace login limits and remaining grace logins.
- If you enable provisioning with the SAML or Liberty protocols, the admin user needs write rights to create users in the user store.
- If you use X.509 authentication, the admin user needs write rights to update the user's login status attributes.

If your user store is an eDirectory user store, Access Manager verifies that the admin user has sufficient rights to browse for users in the specified search contexts.

IMPORTANT: This check is not performed for Active Directory or Sun ONE. If your users cannot log in, you need to verify that you have given the admin for the user store sufficient rights to the specified search contexts.

3.1.4 Configuring a User Store for Secrets

Access Manager allows you to securely store user secrets. These secrets can then be used in Form Fill and Identity Injection policies. Where and how the secrets are stored depends upon your user store and your configuration:

- "Configuring the Configuration Datastore to Store the Secrets" on page 110.

  If you want to do minimal configuration, you can use the configuration datastore on the Administration Console to store the secrets. To increase the security of the secrets, you should configure the security options.
- "Configuring an LDAP Directory to Store the Secrets" on page 112.

If you are willing to extend the schema and add an attribute to your user object on the LDAP directory, you can store the secrets in your LDAP directory.

- "Configuring an eDirectory User Store to Use SecretStore" on page 114.

  If your user store is eDirectory and you have installed Novell SecretStore, you can select to use the SecretStore on your eDirectory server to store the secrets.

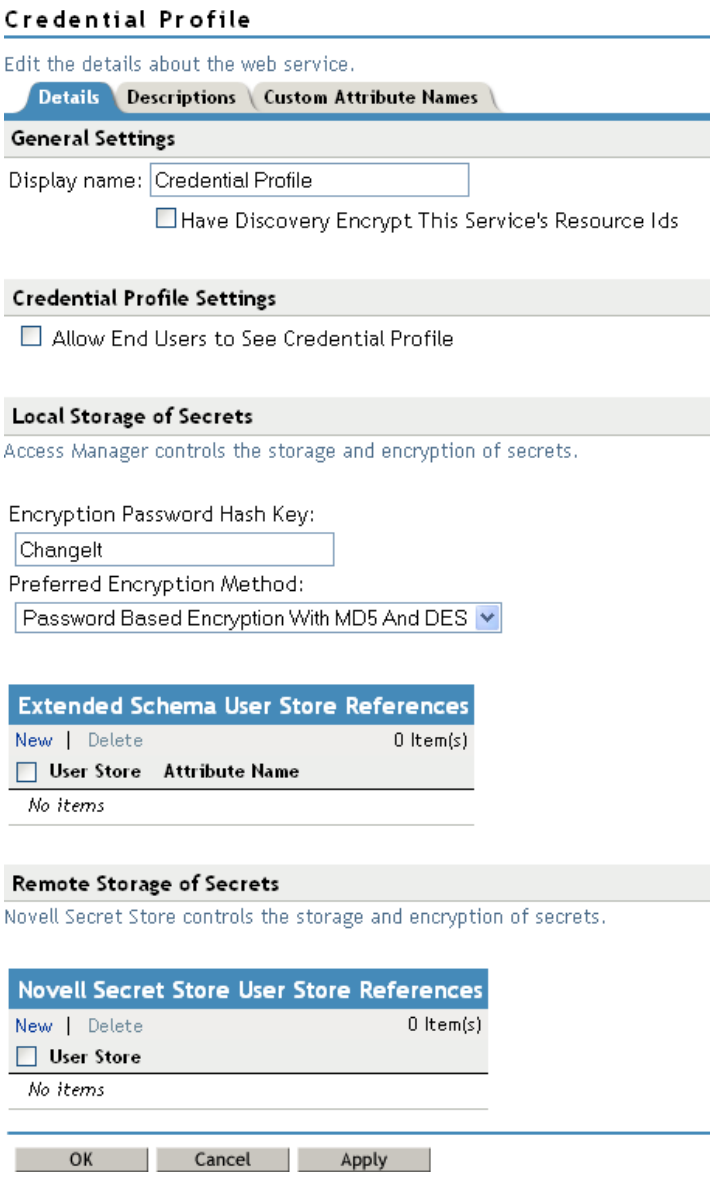
For some troubleshooting tips, see "Troubleshooting the Storing of Secrets" on page 117.

Configuring the Configuration Datastore to Store the Secrets

When you use the configuration datastore of the Administration Console as the secret store, the nidswsfss attribute of the nidsLibertyUserProfile object is used to store the secrets.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.

2 Click *Credential Profile*.



3 Scroll to the *Local Storage of Secrets* section and configure the following security options:

Encryption Password Hash Key: (Required) Specify the password that you want to use as a seed to create the encryption algorithm. To increase the security of the secrets, we recommend that you change the default password to a unique alphanumeric value.

Preferred Encryption Method: Specify the preferred encryption method. Select the method that complies with your security model:

- **Password Based Encryption With MD5 and DES:** MD5 is an algorithm that is used to verify data integrity. Data Encryption Standard (DES) is a widely used method of data encryption that uses a private key.
- **DES:** Data Encryption Standard (DES) is a widely used method of data encryption that uses a private key. Like other private key cryptographic methods, both the sender and the receiver must know and use the same private key.

- **Triple DES:** A variant of DES in which data is encrypted three times with standard DES, using two different keys.

Extended Schema User Store References: Do not specify a user store reference. When this option contains no values, the configuration datastore is used to store the secrets.

4. Click *OK*.
5. On the Identity Servers page, update the Identity Server.
6. To use the secret store to store policy secrets, see "Creating and Managing Shared Secrets" in the *Novell Access Manager 3.1 SP2 Policy Guide*.

Configuring an LDAP Directory to Store the Secrets

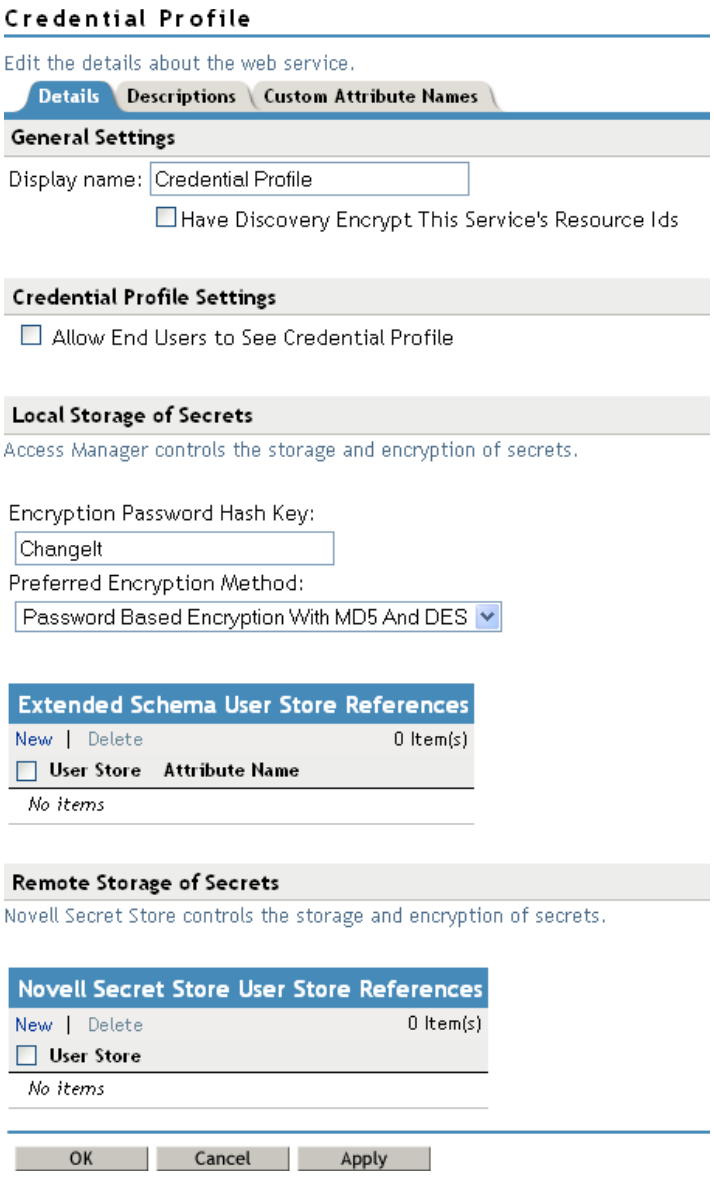
When you use an LDAP directory to store the secrets, you need to enable the user store for the secrets. You select the LDAP directory, then specify an attribute. The attribute you specify is used to store an XML document that contains encrypted secret values. This attribute should be a single-valued case ignore string that you have defined and assigned to the user object in the schema.

To use an LDAP directory to store secrets, your network environment must conform to the following requirements:

- The user class object must contain an attribute that can be used to store the secrets. This attribute must be a string attribute that is single valued and case ignore.
- The user store must be configured to use secure connections (click *Devices* > *Identity Servers* > *Edit* > *Local* > *User Stores* > *[User Store Name]*. In the *Server replicas* section, ensure that the *Port* is 636 and that *Use SSL* is enabled. If they aren't, click the name of the replica and reconfigure it.

To configure the LDAP directory:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Providers*.
2. Click *Credential Profile*.



3 Scroll to the *Local Storage of Secrets* section and configure the following options:

Encryption Password Hash Key: (Required) Specifies the password that you want to use as a seed to create the encryption algorithm. To increase the security of the secrets, we recommend that you change the default password to a unique alphanumeric value.

Preferred Encryption Method: Specifies the preferred encryption method. Select the method that complies with your security model:

- **Password Based Encryption With MD5 and DES:** MD5 is an algorithm that is used to verify data integrity. Data Encryption Standard (DES) is a widely used method of data encryption that uses a private key.
- **DES:** Data Encryption Standard (DES) is a widely used method of data encryption that uses a private key. Like other private key cryptographic methods, both the sender and the receiver must know and use the same private key.

- **Triple DES:** A variant of DES in which data is encrypted three times with standard DES, using two different keys.

4 To specify where to store secret data, click *New* under *Extended Schema User Store References* and fill in the following:

User Store: Select the user store where you want secret store enabled.

Attribute Name: Specify the LDAP attribute that you have created to store the secrets on the selected user store.

5 Click *OK* twice.

6 On the Identity Servers page, update the Identity Server.

7 To create policies that use the stored secrets, see "Creating and Managing Shared Secrets" in the *Novell Access Manager 3.1 SP2 Policy Guide*.

For troubleshooting information, see "Troubleshooting the Storing of Secrets" on page 117.

Configuring an eDirectory User Store to Use SecretStore

For Access Manager to use Novell SecretStore, the user store must be eDirectory and Novell SecretStore must be installed there. When configuring this user store for secrets, Access Manager extends the eDirectory schema for an NMAS method. This method converts authentication credentials to a form understood by eDirectory. For example, Access Manager supports smart card and token authentications, and these authentication credentials must be converted into the username and password credentials that eDirectory requires. This allows the Identity Server to authenticate as that user and access the user's secrets. Without this NMAS method, the Identity Server is denied access to the user's secrets.

To use a remote SecretStore, your network environment must conform to the following requirements:

- The eDirectory server must have Novell SecretStore installed.
- When you configure a user store to use Novell SecretStore, the admin user that you have configured for the user store must have sufficient rights to extend the schema on the eDirectory server, to install the SAML NMAS method, and set up the required certificates and objects. For more information on the rights required, see Section 3.1.3, "Configuring an Admin User for the User Store," on page 109.
- The user store must be configured to use secure connections (click *Access Manager > Identity Servers > Edit > Local > User Stores > [User Store Name]*. In the *Server replicas* section, ensure that the *Port* is 636 and that *Use SSL* is enabled. If they aren't, click the name of the replica and reconfigure it.
- If you have enabled a firewall between the Administration Console and the user store, and between the Identity Server and the user store, make sure that both LDAP ports (389 and 636) and the NCP port (524) are opened.
- If you are going to configure Access Manager to use secrets that are used by other applications, you need to plan a configuration that allows the user to unlock a locked SecretStore. See "Determining a Strategy for Unlocking the SecretStore" on page 116.

To configure the user store:

1 In the Administration Console, click *Devices > Identity Servers > Edit > Local*.

2 Click the name of your user store.

3 Select *Install NMAS SAML method*, then click *OK*.

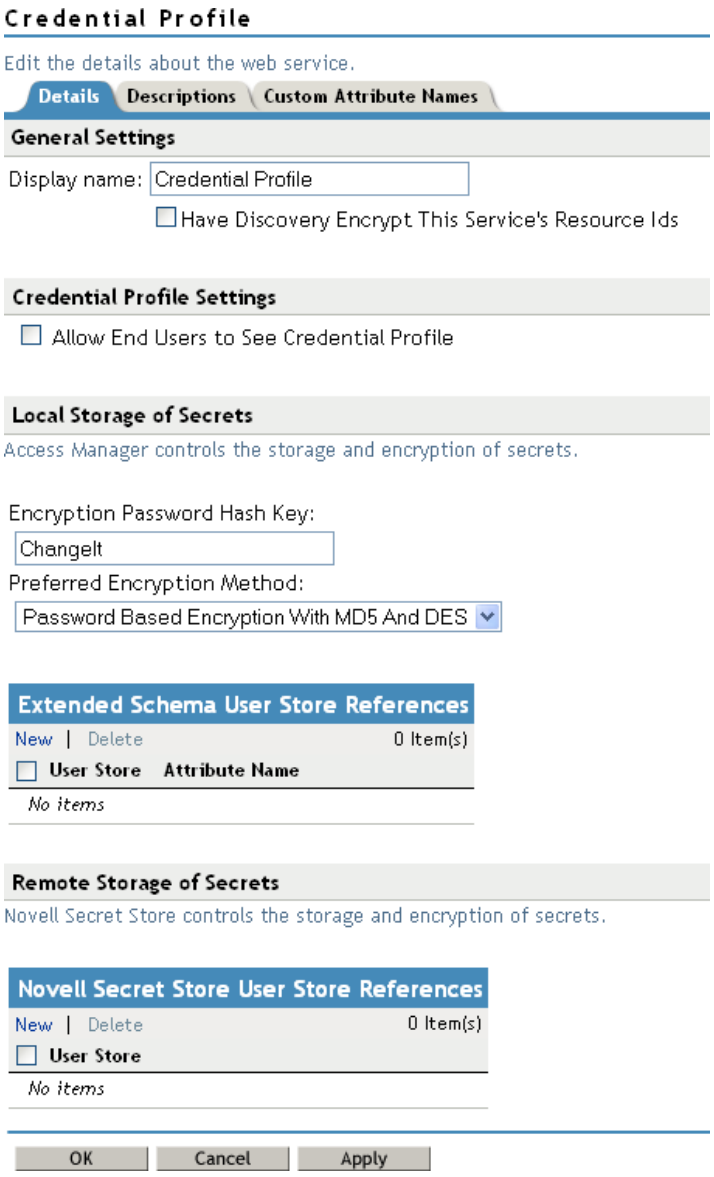
This installs a required NMAS method in the eDirectory schema and adds required objects to the tree.

IMPORTANT: If your eDirectory user store is running on SLES 11 64-bit operating system on x86-64 hardware, the eDirectory server is missing some support libraries that this SAML method requires. For information on installing these libraries, see TID 7006437 (http://www.novell.com/support/viewContent.do?externalId=7006437&sliceId=1).

---

4 Click *Liberty* > *Web Service Providers*.

5 Click *Credential Profile*.



6 Scroll to the *Remote Storage of Secrets* section.

7 Click *New* under *Novell Secret Store User Store References*.

This adds a reference to a user store where SecretStore has been installed.

8 Click the user store that you configured for SecretStore.

9 Click *OK* twice.

10 On the Identity Servers page, update the Identity Server.

11 Continue with one of the following:

- If other applications are using the secret store, you need to determine whether Access Manager users need the option to unlock the secret store. See "Determining a Strategy for Unlocking the SecretStore" on page 116.
- To create policies that use the stored secrets, see "Creating and Managing Shared Secrets" in the *Novell Access Manager 3.1 SP2 Policy Guide*.
- For troubleshooting information, see "Troubleshooting the Storing of Secrets" on page 117.

Determining a Strategy for Unlocking the SecretStore

When an administrator resets a user's password, secrets written to the Novell SecretStore with an enhanced security flag become locked. The Identity Server does not write the secrets that it creates with this flag, but other applications might:

- If Access Manager is not sharing secrets with other applications, the secrets it is using are never locked, and you do not need to configure Access Manager to unlock secrets.
- If Access Manager is sharing secrets with other applications and these application are using the security flag that locks secrets when a user's password is reset, you need to configure Access Manager so that users can unlock their secrets.

If you want users to receive a prompt for a passphrase when secrets are locked, complete the following configuration steps:

1 Require all users to set up a passphrase (also called the Master Password).

Access Manager uses the SecretStore Master Password as the passphrase to unlock the secrets. If the user has not set a passphrase before the SecretStore is locked, this feature of Access Manager cannot unlock the SecretStore. If it is necessary to unlock the SecretStore by using the user's prior password, another tool must be used. See your SecretStore documentation.

2 Configure the Identity Server to perform the check:

2a In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *[User Store Name]*.

2b Select the *Enable Secret Store lock checking* option.

2c Click *OK* twice, then update the Identity Server.

3 Make sure Web Services Framework is enabled:

3a In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Services Framework*.

3b In the *Framework General Settings* section, make sure that *Enable Framework* is selected.

3c Click *OK*. If you made any changes, update the Identity Server.

4 Continue with "Creating and Managing Shared Secrets" in the *Novell Access Manager 3.1 SP2 Policy Guide*.

When the SecretStore is locked and the users log in, the users are first prompted for their login credentials, then prompted for the passphrase that is used to unlock the SecretStore.

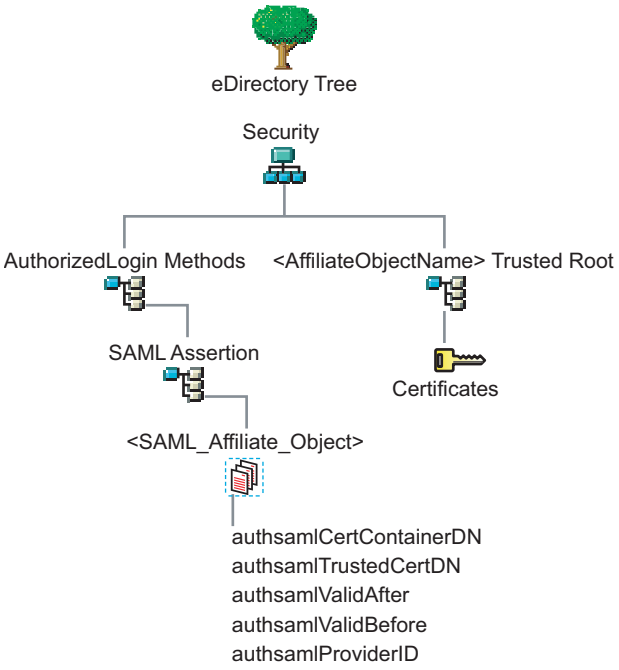
Troubleshooting the Storing of Secrets

Secrets Aren't Stored in Novell SecretStore

When you use Novell SecretStore to store the secrets, the schema on the eDirectory server must be extended, and specific SAML objects and certificates must be created.

To verify that the schema was extended and the objects were created on the eDirectory server:

1. Open an LDAP browser and connect to the eDirectory server.
2. Browse to the Security container.
3. Look for objects similar to the following:



   If the schema has been extended correctly, you can find a SAML Assertion object in the Authorized Login Methods container. The SAML_Assertion object contains an alphanumeric generated name for a SAML affiliate object. This object has four attributes.

   The SAML affiliate object name is used to generate another container in the Security container. This new container is the *<AffiliateObjectName>* Trusted Root container that contains public key signing certificate.

4. Complete one of the following:
   - If these objects do not exist, verify the following, then continue with Step 5:
     - The admin user for the user store has sufficient rights to extend the schema and add these objects to the Security container.

- Any configured firewalls must allow NCP and LDAP traffic for the Administration Console, the Identity Server, and the LDAP user store.
- (Linux) Verify that you have installed the required packages. See "Administration Console Requirements" in the *Novell Access Manager 3.1 SP2 Installation Guide*.

- If the objects exist, check for time synchronization problems. For more information, see "Users Are Receiving Invalid Credential Messages" on page 118.

5 In the Administration Console, modify the secret store configuration so that it is resent to the user store:

5a Click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Providers* > *Credential Profile*.

5b In the *Remote Storage of Secrets* section, remove the user store, then add it again.

5c Click *OK*.

6 On the Identity Servers page, update the Identity Server.

Users Are Receiving Invalid Credential Messages

The <SAML_Affiliate_Object>.SAML-Assertion.AuthorizedLoginMethods.Security object contains two attributes that determine how long credentials are valid. If your Identity Server and eDirectory server are not time synchronized, the credentials can become invalid before a user has time to use them.

Either make sure that the time of your Identity Server and eDirectory server are synchronized, or increase the value of the authsamlValidAfter and authsamlValidBefore attributes of the SAML affiliate object.

Secrets Aren't Stored in the LDAP Directory

1 Open an LDAP browser and connect to the eDirectory server.

2 Browse to the user object.

3 Verify that the user object contains the LDAP attribute that you have specified as the attribute to store the secrets.

4 If the attribute exists, browse to the schema and verify that the attribute has the following characteristics:

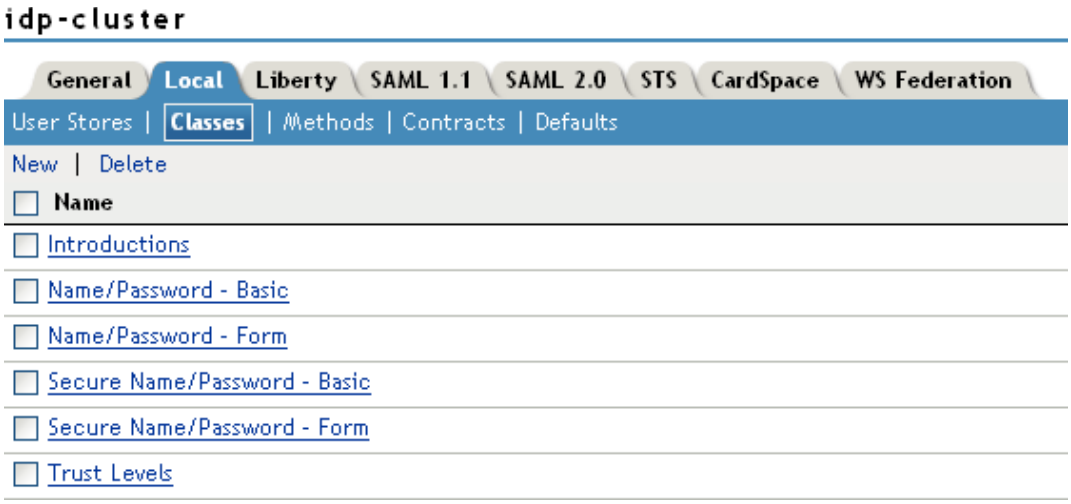
- Single valued
- Case ignore
- String

3.2 Creating Authentication Classes

Authentication classes let you define ways of obtaining end user credentials.You specify the code (Java class) and properties to be executed to implement a particular authentication type.

Several authentication classes are included with Access Manager to provide a variety of ways to authenticate end users. Custom authentication classes provided by other vendors can also be configured to run in the system.

1 In the Administration Console, click *Devices* > *Identity Server* > *Edit* > *Local* > *Classes*.



The following classes are predefined for Access Manager:

Introductions: When the class is configured, it allows users to select an identity provider from a list of introducable identity providers. For information on how to configure and use this class, see Section 7.2.3, “Configuring the Introductions Class,” on page 192.

Name/Password - Basic: Basic authentication over HTTP using a standard login pop-up page provided by the Web browser.

Name/Password - Form: Form-based authentication over HTTP or HTTPS.

Secure Name/Password - Basic: Basic authentication over HTTPS using a standard login page provided by the Web browser.

Secure Name/Password - Form: Form-based authentication over HTTPS.

Trust Levels: When this class is configured, it defines authentication levels for classes that can be used in authentication requests. For more information on how to configure and use this class, see Section 7.2.4, “Configuring the Trust Levels Class,” on page 193.

2 To delete a class, select the class, then click *Delete*.

You cannot delete a class if a method is using it.

For information on how to create a name/password class, see the following sections:

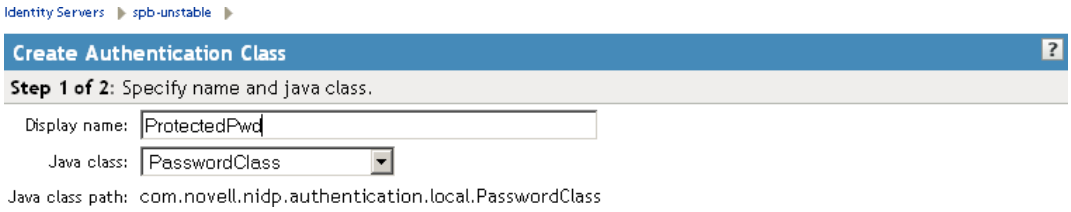
- Section 3.2.1, “Creating Basic or Form-Based Authentication Classes,” on page 120.
- Section 3.2.2, “Specifying Common Class Properties,” on page 121

Some classes require additional configuration to enable their use for authentication. See the following sections:

- “Configuring for RADIUS Authentication” on page 141
- “Configuring Mutual SSL (X.509) Authentication” on page 142
- “Creating an ORed Credential Class” on page 147
- “Configuring for OpenID Authentication” on page 149
- “Configuring Password Retrieval” on page 150
- “Configuring Access Manager for NESCM” on page 153
- “Configuring for Kerberos Authentication” on page 163

3.2.1 Creating Basic or Form-Based Authentication Classes

1. In the Administration Console, click *Devices* > *Identity Server* > *Edit* > *Local* > *Classes*.
2. Click *New* to launch the *Create Authentication Class Wizard*.



3. Specify a display name, then select a class from the *Java class* drop-down menu.
   - The following classes are recommended only for testing purposes:

     BasicClass: Uses basic HTTP authentication.

     PasswordClass: Passes the user name and password over HTTP in readable text, and uses a form-based login to collect the name and password.

     RadiusClass: RADIUS enables communication between remote access servers and a central server. For a production environment, use ProtectedRadiusClass.
   - For a production environment, select one of the following protected classes:

     X509Class: Certificate-based authentication. See Section 4.2, “Configuring Mutual SSL (X.509) Authentication,” on page 142.

     ProtectedBasicClass: The BasicClass, protected by HTTPS.

     ProtectedPasswordClass: The PasswordClass, protected by HTTPS (form-based).

     ProtectedRadiusClass: The RadiusClass, protected by HTTPS. See Section 4.1, “Configuring for RADIUS Authentication,” on page 141 for configuration steps.

     KerberosClass: The authentication class used for using Kerberos for Active Directory and Identity Server authentication. See Section 5, “Configuring for Kerberos Authentication,” on page 163 for configuration steps.

     NMASAuthClass: The authentication class used for Novell Modular Authentication Services (NMAS), which uses fingerprint and other technology as a means to authenticate a user. For instructions on using the NMAS NESCM method, see Section 4.6, “Configuring Access Manager for NESCM,” on page 153.

     NPOrRadiusOrX509Class: The authentication class that allows the creation of a contract from which the user can select an authentication method: name/password, RADIUS, or X.509. For configuration information, see Section 4.3, “Creating an ORed Credential Class,” on page 147.

     PasswordFetchClass: The authentication class that allows the Identity Server to retrieve the user’s password when the user has used a non-password class for authentication. For configuration information, see Section 4.5, “Configuring Password Retrieval,” on page 150.

     OpenIDClass: The authentication class that allows you to configure the Identity Server to trust the provider or providers of OpenIDs. For configuration information, see Section 4.4, “Configuring for OpenID Authentication,” on page 149.

Other: Used for third-party authentication classes or if you have written your own Java class. For information on how to write your own class, see Novell Access Manager Developer Tools and Examples (http://developer.novell.com/wiki/index.php/Novell_Access_Manager_Developer_Tools_and_Examples).

4 Click *Next* to configure the properties for each class. Click *New*, then enter a name and value. The names and values you enter are case sensitive. See Section 3.2.2, "Specifying Common Class Properties," on page 121 for the properties that are used by the basic and password classes.

5 Click *Finish*.

6 Continue with Section 3.3, "Configuring Authentication Methods," on page 123.

To use an authentication class, the class must have one or more associated methods.

3.2.2 Specifying Common Class Properties

The following properties can be used by the basic and password classes:

- "Query Property" on page 121
- "JSP Property" on page 122
- "MainJSP Property" on page 123

These properties can also be specified on a method derived from the class. If you are going to create multiple methods from the same class, consider the following conditions:

- If you want the methods to share the same properties, you can save configuration steps by defining the properties on the class.
- If you want the methods to use different values for the properties such as one method specifying one custom login page and another method specifying a different custom login page, then you should specify the properties on the method.

Query Property

Normally, the Identity Server uses the username to find a user in the user store. You can change this behavior by using the Query property. This property determines the username value for authentication. The default Query string prompts the users for the value of the CN attribute. You can modify this by requesting a different attribute in the LDAP query.

The Query property can be used by the following classes:

- BasicClass
- PasswordClass
- ProtectedBasicClass
- ProtectedPasswordClass

For example, to query for the user's UID attribute to use for the username, you would specify the following query:

Property Name: Query

Property Value: `(&(objectclass=person)(uid=%Ecom_User_ID%))`

The values are case sensitive. The name of the property must be Query with an initial capital. The `%Ecom_User_ID%` variable is used in the default `login.jsp` for the username in the four classes that support the Query property. The variable is replaced with the value the user enters for his or her username, and the LDAP query is sent to the user store to see if the user's attribute value matches the entered value. You can specify any attribute for the Query that is defined in your user store for the object class of person and that is used to identify the user.

The Query you define for the BasicClass and the ProtectedBasicClass needs to use an attribute that your users define as their username. The PasswordClass and the ProtectedPasswordClass do not have this requirement. They also support the JSP property, which allows you to specify a custom login.jsp and have it prompt for other attributes that can be used for login.

For example, you can define the following Query to prompt the users for their email address rather than their username.

Property Name: Query

Property Value: `(&(objectclass=person)(email=%EMail Value%))`

The `%EMail Value%` must match the variable in the custom login page that is filled in when the users enter their credentials. The `objectclass` value must be a valid object class in the LDAP user store. The email attribute must be a valid attribute of the person class.

When you specify such a Query, you must also modify the login page to prompt the user for the correct information. Instead of prompting the user for a username, the login form should prompt the user for an e-mail address. The JSP Property allows you to specify a custom login page. For information on creating a custom login page, see Section 2.1, "Customizing the Identity Server Login Page," on page 59.

JSP Property

The JSP property allows you to specify a custom login page. This property can be used with the following classes:

- PasswordClass
- ProtectedPasswordClass

The property name is JSP and the property value is the filename of the login page you customized without the `.jsp` extension of the file. The property value cannot contain `nidp` in its name.

For example, if you created a custom file named `emaillogin.jsp`, you would specify the following values. The values are case sensitive. The property name needs to be entered as all capitals.

Property Name: JSP

Property Value: `emaillogin`

If you use two methods to create a contract, this property must be set to the same value on both or set on only one. When it is set on only one method, the value is applied to both. This property needs to be used with the MainJSP Property. For information on how to create a custom login page, see Section 2.1, "Customizing the Identity Server Login Page," on page 59.

MainJSP Property

When the MainJSP property is set to true, it indicates that you want to use the page specified in the JSP property for the login page. When this property is set to false, which is the default value, the `nidp.jsp` is used for the login page. If you use two methods to create a contract, this property must be set to the same value on both or set on only one. When it is set on only one method, the value is applied to both.

Property Name: `MainJSP`

Property Value: `true`

For information on how to create a custom login page, see Section 2.1, "Customizing the Identity Server Login Page," on page 59.

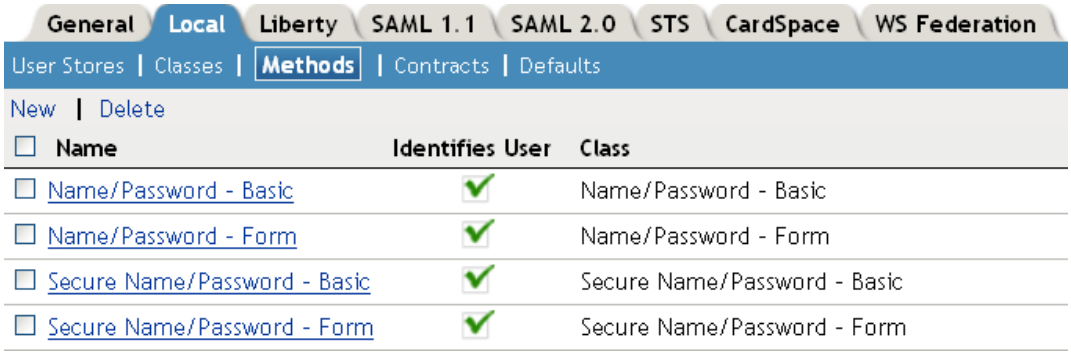
3.3 Configuring Authentication Methods

Authentication methods let you associate authentication classes with user stores. You use a particular authentication class to obtain credentials about an entity, and then validate those credentials against a list of user stores.

After the system locates the entity in a particular user store, no further checking occurs, even if the credentials fail to validate the entity. Typically, the entity being authenticated is a user, and the definition of an authentication method specifies whether this is the case. You can alter the behavior of an authentication class by specifying properties (name/value pairs) that override those of the authentication class.

To configure a method for an authentication class:

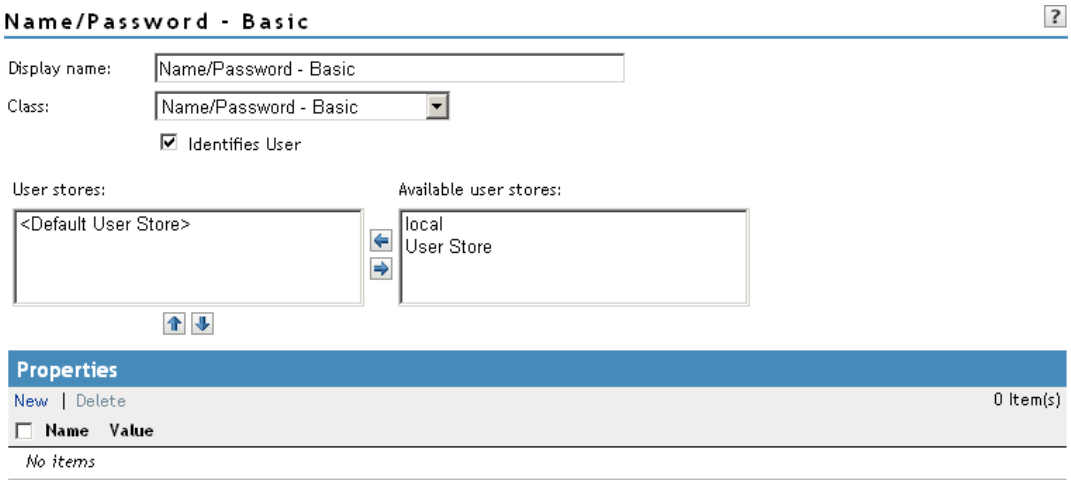
1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Methods*.



2 To delete a method, select the method, then click *Delete*.

A method cannot be deleted if a contract is using it.

3 To modify an authentication method, click its name, or to create one, click *New*.



4 Fill in the following fields:

Display Name: The name to be used to refer to the new method.

Class: The authentication class to use for this method. See Section 3.2, "Creating Authentication Classes," on page 118.

Identifies User: Specifies whether this authentication method should be used to identify the user. Usually, you should enable this option. When configuring multiple methods for a contract, you might need to disable this option for some methods.

If you enable this option on two or more methods in a contract, these methods need to identify the same user in the same user store.

If you enable this option on just one method in the contract, that method identifies the user when the authentication method succeeds. The other methods in the contract must succeed, but might not authenticate the user. For example, the method that identifies the user could require a name and a password for authentication, and the other method in the contract could prompt for a certificate that identifies the user's computer.

5 Add user stores to search.

You can select from the list of all the user stores you have set up. If you have several user stores, the system searches through them based on the order specified here. If a user store is not moved to the *User stores* list, users in that user store cannot use this method for authentication.

<Default User Store>: The default user store in your system. See Section 3.5, "Specifying Authentication Defaults," on page 131.

6 (Optional) Under Properties, click *New*, then fill in the following fields:

Property Name: The name of the property to be set. This value is case sensitive and specific to an authentication class. The same properties that can be set on an authentication class can be set on the method.

You can use the method properties to override the property settings specified on the authentication class. For example, you might want to use the authentication class for multiple companies, but use a slightly different login page that is customized with the company's logo. You can use the same authentication class, create a different method for each company, and use the JSP property to specify the appropriate login page for each company.

For information on the available properties for the basic and form classes, see Section 3.2.2, "Specifying Common Class Properties," on page 121

The Radius classes have the following additional properties that can be set on the method:

- **RADIUS_LOOKUP_ATTR:** Defines an LDAP attribute whose value is read and used as the ID is passed to the RADIUS server. If not specified, the user name entered is used.
- **NAS_IP_ADDRESS:** Specifies an IP address used as a RADIUS attribute. You might use this property for situations in which service providers are using a cluster of small network access servers (NASs). The value you enter is sent to the RADIUS server.

Property Value: The values associated with the *Property Name* field.

7 Click *Finish*.

8 Continue with Section 3.4, "Configuring Authentication Contracts," on page 125.

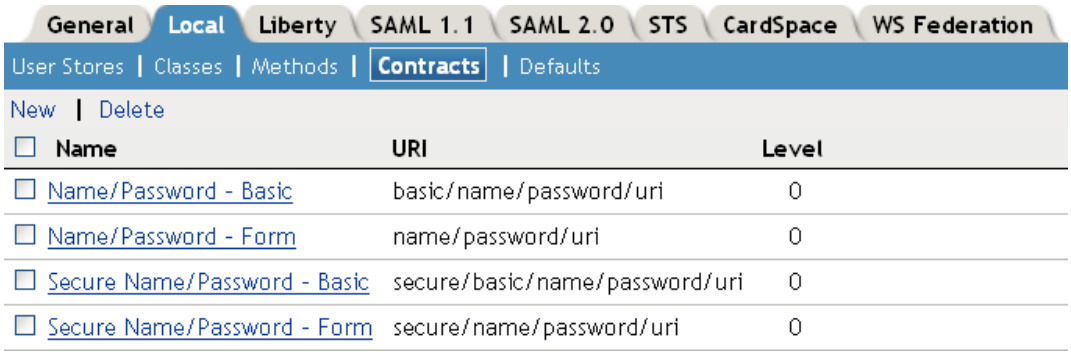
To use a method for authenticating a user, each method must have an associated contract.

3.4 Configuring Authentication Contracts

Authentication contracts define how authentication occurs. An Identity Server can have several authentication contracts available, such as name/password, X.509, or Kerberos. From the available contracts, you assign a contract to a specific resource or resources. It is access to a resource that triggers the authentication process. If the user has already supplied the required credentials for the contract, the user is not prompted for them again.

Each contract is assigned a URI that uniquely identifies it. This URI can be shared with other providers so that they can identify the type of credentials the Identity Provider is requiring. You can also restrict a contract so that it can only be used for local authentication and not with other providers.

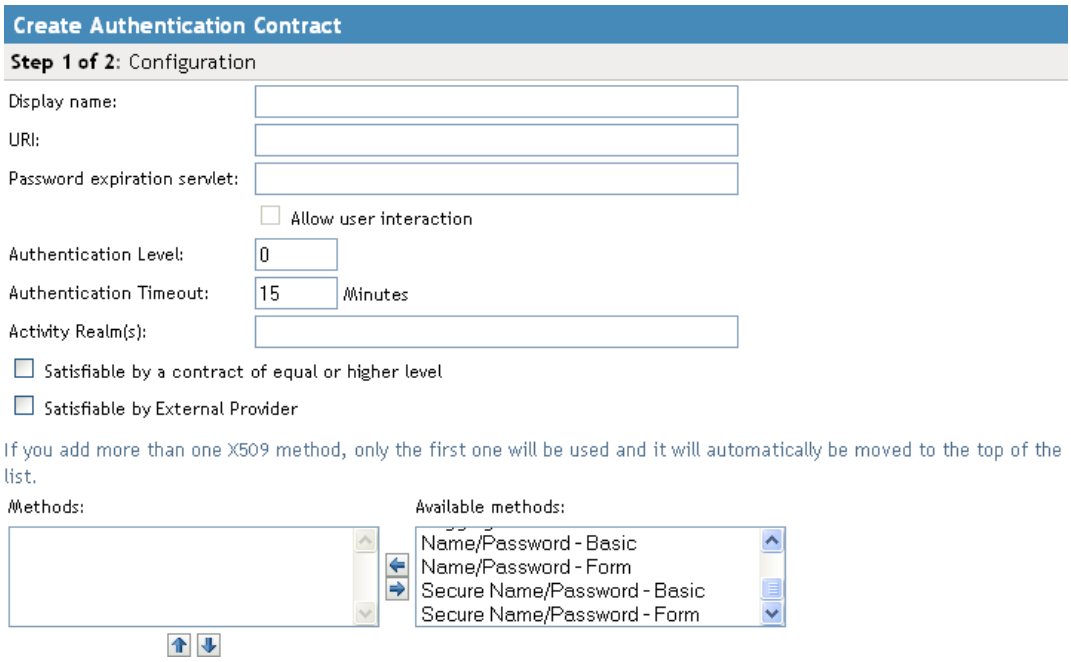
1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts*.



2 To delete a contract, select the contract, then click *Delete*.

You cannot delete a contract if it is in use by an Access Gateway or J2EE agent.

3 To create a new contract, click *New*.



4 Fill in the following fields:

Display name: Specifies the name of the authentication contract.

URI: Specifies a value that uniquely identifies the contract from all other contracts. It is used to identify this contract for external providers and is a unique path value that you create. No spaces can exist in the URI field.

The following are all valid values for the URI:

```
/mycompany/name/password/form
http://mycompany.com/login
secure/form/password/bcompany
```

Password expiration servlet: Specifies a URL to a page where the user can change his or her password. This applies only to eDirectory servers when the password is expired or is within the grace login period. You must use eDirectory to change the number of grace logins.

For more information about how use this type of servlet, see Section 3.4.1, “Using a Password Expiration Service,” on page 128.

Allow User Interaction: If you specify a password expiration servlet, you can enable this option, which allows the users to decide whether to go to the servlet and change their passwords or to skip the servlet. If you always want to force the users to go the servlet to change their passwords, do not enable this option.

Authentication Level: A number you can assign to this authentication contract to specify its security level or rank. You use this setting to preserve authentication contracts of a higher security level. When you enable the *Satisfiable by a contract of equal or higher level* option on this page, the system uses this value as a reference.

For example, you might create a name/password authentication contract and assign it to level one. You might also create an X.509 authentication contract and assign it to level two. If a user supplies the credentials for the X.509 level-two contract, the system does not require the credentials to satisfy the name/password level-one authentication contract.

Authentication Timeout: Specify how long the session can be inactive before the user is prompted to log in again. The value can be from 5 minutes to 66535 minutes and must be divisible by 5.

If you modify the timeout value for a contract, the newly assigned value is given to users as they log in. Currently logged in users retain the old value until they re-authenticate.

You need to experiment to discover what values are best for your network configuration, your security requirements, and your users.

- Shorter timeouts increase back-channel traffic and require more threads for authentication checks, but quickly free resources that are being used by inactive users. If you have slow back-end services, users could get disconnected waiting for a response, and these disconnects can generate more authentication traffic.
- Longer timeouts, which allow inactive users to remain connected, increase memory requirements to store session information, but require fewer threads and don't generate as much back-channel traffic.

For example, if you set the timeout to 5 minutes, an authentication check needs to be done 12 times each hour for each user authenticating with this contract. If the timeout is set to 60 minutes, an authentication check is done only one time each hour for each user. However, for the 5 minute timeout, resources can be freed within 5 minutes of inactivity by the user. For the 60-minute timeout, resources can take as long as 60 minutes to be freed, depending upon when the user goes inactive.

For information on how to use this feature with the Access Gateway, see "Assigning a Timeout Per Protected Resource" in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*

Activity Realm(s): Specify the name of the realm that can be used to indicate activity. Use a comma-separated list to specify multiple realms. This allows a user's session to be kept alive when the user is accessing resources that are protected by different contracts. If both contracts belong to the same realm, activity on either resource keeps the session alive on the other resource. For more information about this feature, see Section 3.4.2, "Using Activity Realms," on page 130.

Satisfiable by a contract of equal or higher level: Allows the system to satisfy this authentication contract if a user has logged in using another contract of an equal or higher authentication level, as specified in the *Authentication Level* field of an authentication contract.

When you enable this option, you need to be aware of the authentication levels you have set for other contracts and the level that has been assigned to the default contract.

Satisfiable by External Provider: Allows this contract to be selected when configuring an identity provider for Liberty or SAML 2.0. When you configure the authentication request, you can select a contract that has this option enabled and require the identity provider to use this contract in order for authentication to succeed.

Methods and Available Methods: Specifies the authentication method to use for the contract. You can specify the order in which the methods are executed for login; however, this is not a graded list, so all the methods you specify are required. *Available methods* are the authentication methods you have set up.

If you add more than one X.509 method, only the first one is used and it is automatically moved to the top of the list.

When you choose a secure method, such as Secure Name/Password, ensure that you have enabled security for the Identity Server configuration by setting the protocol to HTTPS. See "Configuring Secure Communication on the Identity Server" in the *Novell Access Manager 3.1 SP2 Setup Guide*.

5 Click *Next*.

6 Configure a card for the contract by filling in the following:

ID: (Optional) Specify an alphanumeric value that identifies the card. If you need to reference this card outside of the Administration Console, you need to specify a value here. If you do not assign a value, the Identity Server creates one for its internal use.

Text: Specify the text that is displayed on the card to the user.

Image: Specify the image to be displayed on the card. Select the image from the drop down list. To add an image to the list, click *Select local image*.

Show Card: Determine whether the card is shown to the user, which allows the user to select and use the card for authentication. If this option is not selected, the card is only used when a service provider makes a request for the card.

Passive Authentication Only: Select this option if you do not want the Identity Server to prompt the user for credentials. If the Identity Server can fulfill the authentication request without any user interaction, the authentication succeeds. Otherwise, it fails.

7 Click *Finish*, then *OK*.

8 Update the Identity Server and any devices that use the Identity Server configuration.

9 To use this contract, you must configure Access Manager to use it:

- You can assign it as the default contract for the Identity Server. See Section 3.5, "Specifying Authentication Defaults," on page 131.
- You can configure a protected resource to use it. See "Configuring Protected Resources" in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*.

3.4.1 Using a Password Expiration Service

Access Manager works with any password management service that works with your user store. For an implementation example, see Configuring Access Manager for UserApp and SAML (http://www.novell.com/coolsolutions/appnote/19981.html).

As you configure the service, be aware of the following configuration options:

URL Parameters

When you are defining the URL for the password service on the Contracts page, the following optional tags can be used in the parameter definitions of the URL. You need to use parameter names that are understood by the service you have selected to use. The Identity Server does not need to understand these parameters, but the password expiration service needs to understand them.

The table below lists a few common ones. Your service might or might not use these, and might require others.

Parameter	Description
<USERID>	Provides the DN of the user with a password that is expired or expiring.
<STOREID>	Provides the name of the user store that authenticated the user before redirecting the user to the password expiration service.
<RETURN_URL>	Provides the URL at the Identity Server to which the user can be redirected after the password service completes.
action=expire	Causes the password expiration service to behave as though the user's password policy is set to allow the user to reset the password even though the user's policy might be set to show the user a hint. The user sees the page to create a new password rather than seeing a hint for an existing password.

For example:

```
https://someservice.com/path/password?user=<USERID>&store=<STOREID>
&returl=<RETURN_URL>&action=expire
```

NOTE: If you copy and paste this text, make sure you remove the white space between <STOREID> and &returl.

The Identity Server fills in these values, which results in the following URL:

```
https://someservice.com/path/password?user=joe.novell&store=userstore1&
returl=https://myidp.com/nidp/idff/sso&action=expire
```

Forcing Authentication after the Password Has Changed

The password service can also include parameters on the return URL sent to the Identity Server. The Identity Server understands the following parameter:

Parameter	Description
forceAuth=TRUE	When the user is returned to the Identity Server, this parameter forces the user to authenticate with the new password. This eliminates the possibility of an old password being used in an Identity Injection policy.

The following example sends this parameter with https://testnidp.novell.com:8443 as the base URL of the Identity Server.

```
<form id="externalForm" action='https://testnidp.novell.com:8443/nidp/idff/
sso?sid=0&id=117&forceAuth=TRUE' method="post">
```

When the user is redirected to the password management service URL because of an expired password, the POST data in that redirect contains the sid=<> and id=<> values as part of the value used for the Identity Server return URL.

Grace Logins

If you specify a password service and do not specify a value for the number of grace logins in eDirectory, the contract redirects to the password management service only when the grace login count has reached 0 and the password has expired.

The Identity Server needs to read the value of the grace login attribute in order to properly redirect to the password management servlet. If restricting grace logins is not important to your security model, enable grace logins and set the maximum to 9999 (the equivalent of infinite in most environments). For more information, see TID 3465171 (http://www.novell.com/support/php/search.do?cmd=displayKC&docType=kc&externalId=3465171&sliceId=2&docTypeID=DT_TID_1_1&dialogID=131458644&stateId=0%200%20131454892).

Federated Accounts

A user's password does not expire and grace logins are not decremented when you have the following setup:

- The Identity Server is configured to act as a service provider
- User identification is configured to allow federation
- Federation is set up with SAML 2.0, Liberty, WS Federation, or CardSpace protocols

The password expiration service is not called because the user is not using a password for authentication. The service can only be called when the user's account is defederated. After the user has defederated the account, the next time the user logs in, a password is required and the service is called.

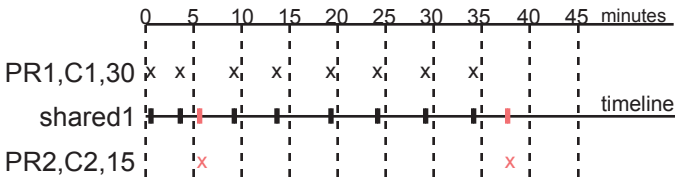
3.4.2 Using Activity Realms

Activity realms are designed to be used with an Access Manager system that uses multiple contracts to protect resources that require different activity timeouts. Activity realms allow you to define how activity at one protected resource affects the activity timeout at another protected resource.

An activity realm essentially represents a time line that tracks the last activity for any resource that is protected by a contract assigned to the activity realm. When a protected resource is accessed, the activity realm associated with the contract is marked as having activity. The contract times out for a protected resource when the elapsed time for activity on the activity realm is greater than the time limit specified in the contract.

For example, suppose you create an activity realm called shared1 and assign it to contract C1 with a timeout of 30 minutes and to contract C2 with a timeout of 15 minutes. Any activity at the resource protected by C1 or C2 marks activity to the shared1 time line. Figure 3-3 illustrates this scenario.

Figure 3-3 *Two Contracts Sharing an Activity Realm*



In Figure 3-3, the user logs into PR1 at time 0, then logs into PR2 at time 6. During the next 30 minutes, the user is active on PR1. The time line for the shared1 activity realm is updated with the user's activity. The user then access PR2 at time 38. Even though no activity has taken place on PR2 for more than the 15-minute contract timeout, PR2 does not time out because activity has occurred within this time at PR1 and because the resources share the same activity realm. Assigning two or more contracts to the same activity realm allows the contracts to influence the timeouts of the other contracts in the activity realm.

When you configure protected resources to use different contracts with different timeouts, they can keep each other alive when they share the same activity realm. If protected resources should not affect each other's activity, they must not share a common activity realm.

You can assign a contract to multiple activity realms. With this configuration, activity on a resource updates the time lines of all activity realms associated with the contract. As long as one of the activity realms has activity within the contract's timeout limit, the user's session remains authenticated.

Activity realms are defined by specifying a name, and the names are case insensitive. Use a comma-separated list to specify multiple names. The system has two default realms that you can use:

- **Any:** Leave the field blank or specify `any` when you want the user's session to remain alive as long as there is some activity by the user at the Access Gateway or at the Identity Server.

  When the Identity Server receives an assertion from another Identity Server that cannot be mapped to a contract, the activity realm is set to `any` with the timeout value equal to the value of the Tomcat session. (The Tomcat session timout is set to the greatest timeout value of the contracts configured for the Identity Server.)
- **NIDPActivity:** Specify `NIDPActivity` for the realm when any activity at the Identity Server by the user can be used to keep the user's session alive.

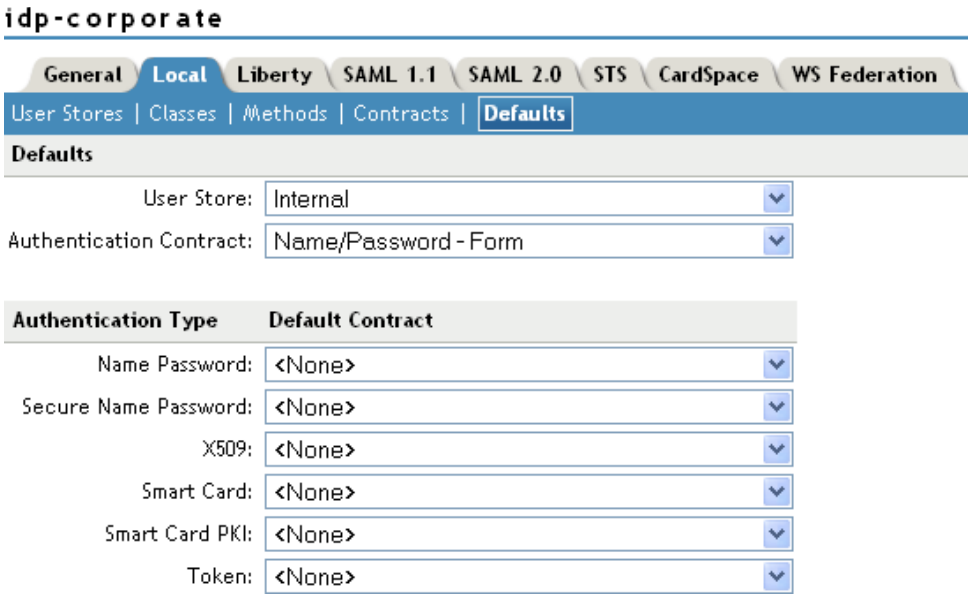
When you place multiple contracts in the same activity realm, you need to plan carefully so that security limits aren't overruled by activity on less critical protected resources. You also need to carefully balance the desire for single sign-on with the need to require reauthentication for sensitive data. Highly sensitive resources are most secure when they are protected by a contract that is created from its own unique method and that is assigned its own unique activity realm. For more information, see "Assigning a Timeout Per Protected Resource" in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*.

3.5 Specifying Authentication Defaults

You can specify default values for how the system processes user stores and authentication contracts. The default contract is executed when users access the system without a specified contract, and when the Access Gateway is configured to use any authentication.

Additional default contracts can be specified for well-known authentication types that might be required by a service provider. These contracts are executed when a request for a specific authentication type comes from a service provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Defaults*.



2 Configure the following fields as necessary:

User Store: Specifies the default user store for local authentication. If you selected *<Default User Store>* when configuring an authentication method, the system uses the user store you specify here.

Authentication Contract: Specifies the default authentication contract to be used when users access the Identity Server directly or a protected resource is configured to use *Any Contract*. If you create a new contract and specify it as the default, ensure that you update the Access Gateway configuration if it has protected resources configured to use *Any Contract*. See "Configuring Protected Resources" in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*.

Authentication Type: Specifies the default authentication contracts to be used for each authentication type. When a service provider requests a specific authentication type, rather than a contract, the identity provider uses the authentication contract specified here for the requested authentication type. For more information, see Section 3.5.1, "Specifying Authentication Types," on page 132.

3 Click *OK*.

4 Update the Identity Server.

3.5.1 Specifying Authentication Types

Trusted service providers can send the Identity Server an authentication request that contains a request for contract or for an authentication type. When the request is for an authentication type, the Identity Server must translate the type to a contract before authenticating the user. You can use the *Authentication Type* section of the Defaults page to specify which contract to use for the common types (classes).

The Identity Server has not implemented all possible types. For types that do not appear on the Defaults page, you can do one of the following:

- You can define a contract for the class whose URI matches the requested class type. When the authentication request is received, the Identity Server uses the URI to match the request with a contract.

  When you create such a contract, you are stating that the contract is security equivalent to the class that is being requested. For configuration information, see Section 3.5.2, "Creating a Contract for a Specific Authentication Type," on page 133.

- You can use the Trust Levels class to assign an authentication level for the requested class. This level is used to rank the requested type. Using the authentication level and the comparison context, the Identity Server can determine whether any contracts meet the requirements of the request. If one or more contracts match the request, the user is presented with the appropriate authentication prompts.

  For configuration information, see Section 7.2.4, "Configuring the Trust Levels Class," on page 193.

3.5.2 Creating a Contract for a Specific Authentication Type

The following steps explain how to create a contract that matches what a trusted service provider is asking for in its authentication request.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts*.
2. To create a new contract, click *New*.
3. Fill in the following fields:

   Display name: Specifies the name of the authentication contract.

   URI: Specifies a value that uniquely identifies the contract from all other contracts. This value must match what the service provider is sending in its authentication request for the type.

   Authentication Level: (Optional) Specify a security level or rank for the contract. This value is not used when authentication request sets the comparison type to exact. It is only used when a contract is selected based on a comparison of authentication levels.

   If the service provider sets the comparison type to minimum, the authentication level can be the same or higher. If the comparison type is set to better, the authentication level must be higher.

   Methods: Select the method that matches the class or type you specified in the URI.

   The other fields for the contract are not requirements of the authentication request and can be configured to meet the requirements of the Identity Server. For information about these fields, see Section 3.4, "Configuring Authentication Contracts," on page 125.
4. Click *Next*.
5. Configure an authentication card for the contract.

   For information about these fields, see Section 3.4, "Configuring Authentication Contracts," on page 125.
6. Click *Finish*, then *OK*.
7. Update the Identity Server.

3.6 Managing Direct Access to the Identity Server

Users usually log into the Identity Server when they request access to a Web resource. They are redirected by the Access Gateway from the resource to the Identity Server to provide the required credentials for the resource. After they are authenticated, they are not prompted for credentials again, unless a resource requires credentials that they haven't already supplied.

However, users can log directly into the Identity Server and access the User Portal, or they can access information about available Web Services Description Language (WSDL) services. This section describes how to manage access to these pages.

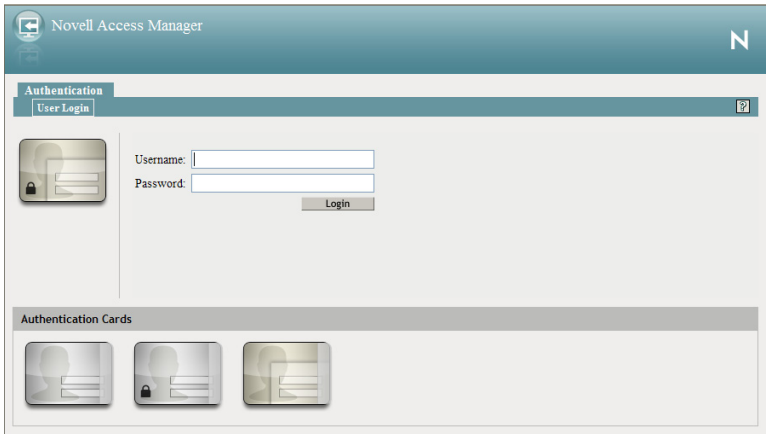
3.6.1 Logging In to the User Portal

Users can log directly in to the Identity Server when they enter the Base URL of the Identity Server in their browsers. For example, if your base URL is `http://doc.provo.novell.com:8080/nidp`, users can log in directly to the Identity Server by entering the following URL:

```
http://doc.provo.novell.com:8080/nidp/app
```

This URL prompts the user to authenticate with the credentials required for the default contract.

Figure 3-4 *User Portal*



When users log directly into the Identity Server, the users need to use the default card for authentication. This is the card that appears in the top left frame, and the credentials it requires are displayed in the top right frame.

On a newly installed system, cards for all the authentication contracts that are installed with the system are displayed. To avoid confusing your users, you need to disable the *Show Card* option for the contracts you do not want your users to use. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts* > *[Name of Contract]* > *Authentication Card*.

Also, make sure you modify the default contract to match a card that is displayed. In the Administration Console, click *Devices > Identity Servers > Edit > Local > Defaults*.

If you display multiple cards, users can use different credentials to authenticate multiple times by selecting another authentication card and entering the required credentials. This is only useful if the credentials grant the user different roles or authorize access to different resources.

If you have configured the Identity Server to be a service provider and have established a trusted relationship with one or more identity providers, the cards of these trusted identity providers appear in the *Authentication Cards* section. Your users can use the identity provider's authentication card to federate their account at the identity provider with their account at the service provider. When they federate an account, they are telling the service provider to trust the authentication established at the identity provider. This enables single sign-on between the providers. The card can also be used to defederate the accounts. On the authentication card, click *Card Options*, then select *Defederate*.

If you have configured the Identity Server to be an identity provider for service providers, a Federation page is accessible after login. From this page, users can federate and defederate their accounts with trusted service providers.

3.6.2 Specifying a Target

You need to specify a target for the following conditions:

- You want to direct the users to a specific URL after the users log in to the Identity Server.
- You do not want users to have access to the User Portal page.

Use one of the following methods to specify the target:

- **Specify a Target in the URL:** You can have your users access the Identity Server with a URL that contains the desired target. For example:

  `https://<domain.com>:8443/nidp/app?target=http://www.novell.com`

  where <*domain.com*> is the DNS name of your Identity Server. In this example, the users would see the Novell Web site after logging in.

- **Specify a Hidden Target on your Form:** If you have your own login form to collect credentials and are posting these credentials to the Identity Server, you can add a hidden target to your login form. When authentication succeeds, the user is directed to this target URL. This entry on your form should look similar to the following:

  ```
  <input type="hidden" target="http://www.novell.com">
  ```

These methods work only when the user's request is for the `/nidp/app`. If the user's request is a redirected authentication request for a protected resource, the protected resource is the target and cannot be changed.

3.6.3 Blocking Access to the User Portal Page

If a user is already authenticated and accesses the Identity Server, the user is presented with the Identity Server User Portal page.

This page provides a wealth of information about the logged-in user:

- Any federations this user has established with third-party service providers
- Identity attributes such as Liberty Personal or employee profile attributes, or Access Manager credential or custom profile attributes
- Policy attributes that users or administrators have selected to share with other service providers

You might want to prevent users from seeing this page for the following reasons:

- **Security:** Users accessing this page have access to sensitive information that administrators might want to restrict such as the user's attributes and federations with other third-party SAML or Liberty providers.
- **Help Desk Support:** Most users have no need to access the information presented in this page. As a result, they might be confused if they see it. By preventing access to the page, any potential calls into the help desk are avoided.

The `main.jsp` page is called with every access to the Identity Server login page. You can modify the code that checks the users status, and if the user is already authenticated, you can redirect the user to another page.

To block access to the User Portal page:

1. Open the `main.jsp` file for editing. This file is located in the following directory:

   Linux: `/var/opt/novell/tomcat5/webapps/nidp/jsp`

   Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\jsp`

   Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\jsp`
2. Locate the following line:

   ```
   ContentHandler hand = new ContentHandler(request,response);
   ```

3 Add the following lines just below this line:

```
<%
if (hand.isAuthenticatedSession())
{
    String redirectURL = "http://www.novell.com/";
    response.sendRedirect(redirectURL);
}
%>
```

Replace the redirectURL value ("`http://www.novell.com/`") with the URL you want your users redirected to.

When a user accesses the login page and is not authenticated, the login process continues with its default process, and the user is presented with the login page where the user credentials can be entered and submitted. If the user is already logged in, the `isAuthenticatedSession()` function returns true. Instead of being redirected to the default IDP portal page, the new code is executed, and the user is redirected to a predefined URL.

The following ieHTTPHeaders output confirms this:

```
GET /nidp/app HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/
x-shockwave-flash, application/x-ms-application, application/x-ms-xbap,
application/vnd.ms-xpsdocument, application/xaml+xml, */*
Accept-Language: en-US,en-IE;q=0.5
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022; .NET CLR
3.0.4506.2152; .NET CLR 3.5.30729)
Host: idp126.lab.novell.com:8443
Connection: Keep-Alive
Cookie: JSESSIONID=11AB34250B3E79DEC11186168C23B34D; novell_language=en-
us; CoreID6=2349599598221244044 9949;
__utma=64695856.419410920.1252432782.1270822885.1271090179.10;
__utmz=64695856.1270722077.8.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(no
ne); WT_FPC=id=83.147.135.44-
1904004976.30060919:lv=1266928072031:ss=1266927852968; WT_DC=tsp=1;
IPCZQX03a36c6c0a=000002009302249462bb469a9f0f5b43243b858a
HTTP/1.1 302 Moved Temporarily
Server: Apache-Coyote/1.1
Pragma: No-cache
Cache-Control: no-cache
Location: http://www.novell.com/
Content-Type: text/html;charset=UTF-8
Content-Length: 0
Date: Thu, 29 Apr 2010 09:17:19 GMT
```

4 Copy this modified `main.jsp` file to each Identity Server in the cluster.

5 Make a backup copy of this file. Whenever you upgrade the Identity Server, this file is overwritten.

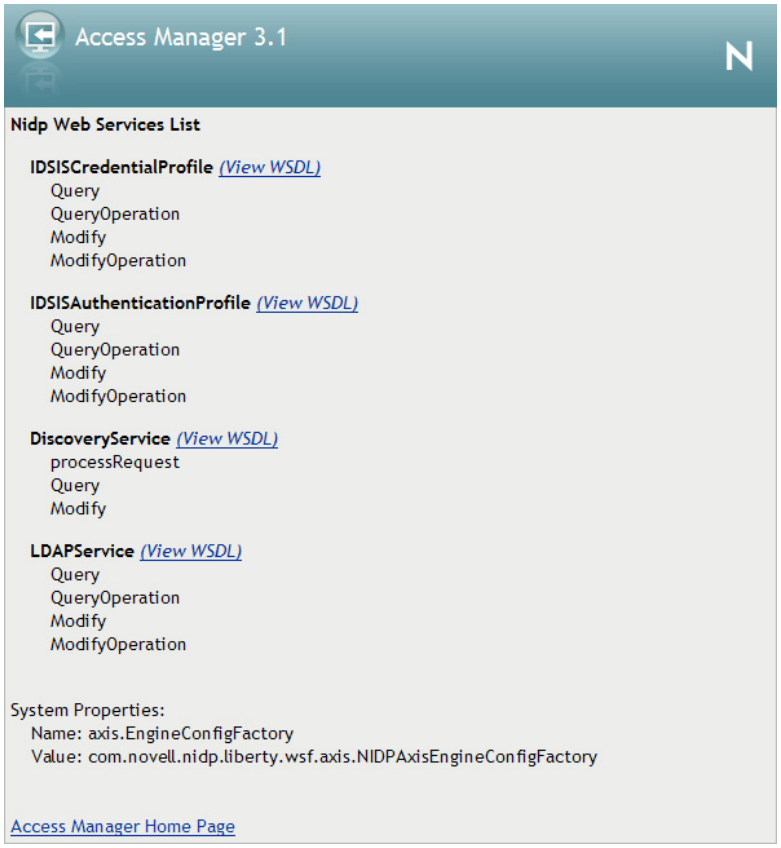
3.6.4 Blocking Access to the WSDL Services Page

Users can access the WSDL services page when they enter the base URL of the Identity Server in their browsers with the path to the Services page. For example, if your base URL is `http://bfrei.provo.novell.com:8080/nidp`, the users can access the services page with the following URL:

`http://bfrei.provo.novell.com:8080/nidp/services`

The Services page contains the following information and links:

Figure 3-5 *WSDL Services Page*



The amount of information displayed on this page depends upon the profiles you have enabled. To enable profiles, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.

If you do not want your users to have access to this page, you can block access.

1. Log in as the root or administrator user.
2. Open the `web.xml` file for editing:

   Linux: `/opt/novell/nids/lib/webapp/WEB-INF`

   Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\WEB-INF`

   Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF`
3. Near the top of the file, in the context initialization parameters section, add the following lines:

   ```
   <context-param>
     <param-name>wsfServicesList</param-name>
     <param-value>full</param-value>
   </context-param>
   ```

   When `<param-value>` has a value of `full`, users can access the Services page. To modify this behavior, replace `full` with one of the following values:

Value	Description
404	Returns an HTTP 404 status code: Not Found
403	Returns an HTTP 403 status code: Forbidden
empty	Returns an empty services list

If the parameter is removed from the file or if you enter an invalid value, the value is interpreted as `full`, and users have access to the page.

4 Restart Tomcat for your modifications to take effect:

Linux: Enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

Windows: Enter the following commands:

```
net stop Tomcat5
net start Tomcat5
```

Configuring Advanced Local Authentication Procedures

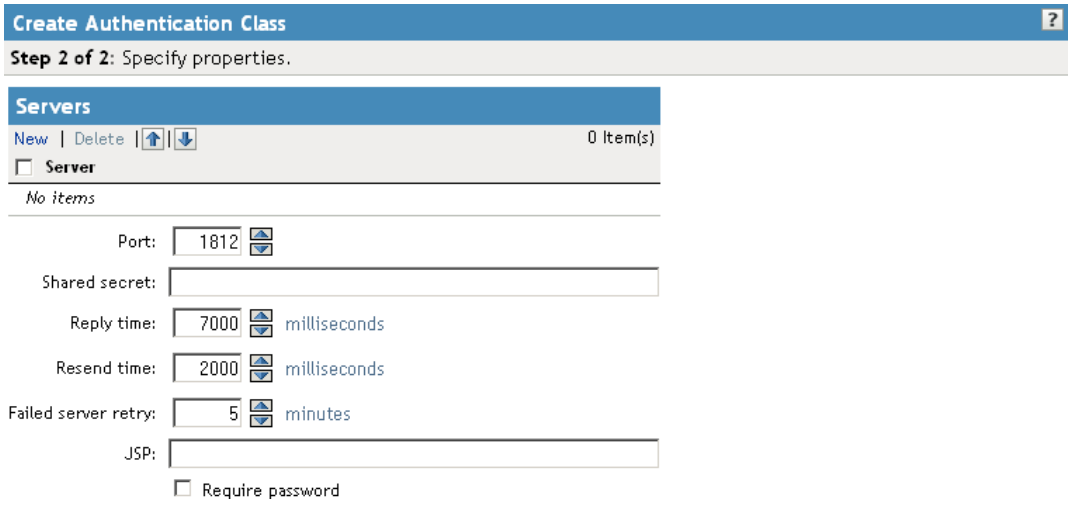
4

The following authentication procedures require more than a username and password. Some of them require that you configure another server to provide the user with a token or a certificate.

4.1 Configuring for RADIUS Authentication

RADIUS enables communication between remote access servers and a central server. Secure token authentication through RADIUS is possible because Access Manager works with Novell Modular Authentication Service (NMAS) RADIUS software that can run on an existing NetWare server. Access Manager supports both PIN and challenge-and-response methods of token-based authentication. In other words, RADIUS represents token-based authentication methods used to authenticate a user, based on something the user possesses (for example, a token card). Token challenge-response is supported for two-step processes that are necessary to authenticate a user.

1. In the Administration Console, click *Devices* > *Identity Server* > *Edit* > *Local* > *Classes*.
2. Click *New*.
3. Specify a display name, then select *RadiusClass* or *ProtectedRadiusClass* from the drop-down menu.
4. Click *Next*.



5 Click *New* to add an IP address for the RADIUS server. You can add additional servers for failover purposes.

6 Click *OK*.

7 Fill in the following fields:

Port: The port of the RADIUS server.

Shared Secret: The RADIUS shared secret.

Reply Time: The total time to wait for a reply in milliseconds

Resend Time: The time to wait in milliseconds between requests.

Server Failure Retry: The time in milliseconds that must elapse before a failed server is retried.

JSP: Specify the name of the login page if you want to use something other than the default page. The filename must be specified without the JSP extension. The default page is used if nothing is specified.

Require Password: Select to require the user to also specify an LDAP password.

8 Click *Finish*.

9 Create a method for this class.

For instructions, see Section 3.3, "Configuring Authentication Methods," on page 123.

10 Create a contract for the method:

For instructions, see Section 3.4, "Configuring Authentication Contracts," on page 125.

If you want the user's credentials available for Identity Injection policies and you did not enable the *Require Password* option, add the password fetch method as a second method to the contract. For more information about this class and method, see Section 4.5, "Configuring Password Retrieval," on page 150.

11 Update the Identity Server.

4.2 Configuring Mutual SSL (X.509) Authentication

Mutual authentication is used when a user is issued an X.509 certificate from a trusted source, and the certificate is then used to identify the user. To ensure the validity of the certificates, Access Manager supports both Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP) methods of verification.

In order for X.509 authentication to succeed, the LDAP admin user defined for the user store must have read and write rights to the LDAP server. When a user tries to authenticate by using the X.509 authentication, an LDAP lookup is performed to identify the user based on the certificate attributes defined in the authentication class. If the user is identified, an LDAP login policy check is performed by using LDAP extensions. Because there is no user password with X.509 certificates, the check is performed by using the admin user defined for the user store. This check is done to update the user login status which requires the write right.

To configure X.509 authentication, you need to create an authentication class, then configure the validation and attribute mapping options.

1 Log in to the Administration Console.

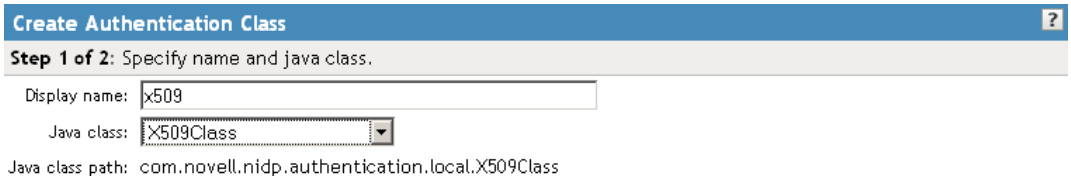
2 Import the trusted root certificate or certificate chain of the Certificate authority into the Identity Server trusted root store.

For information on how to import trusted roots, see "Importing Public Key Certificates (Trusted Roots)" in the *Novell Access Manager 3.1 SP2 Administration Console Guide*.

The Identity Server must trust the Certificate authority that created the user certificates.

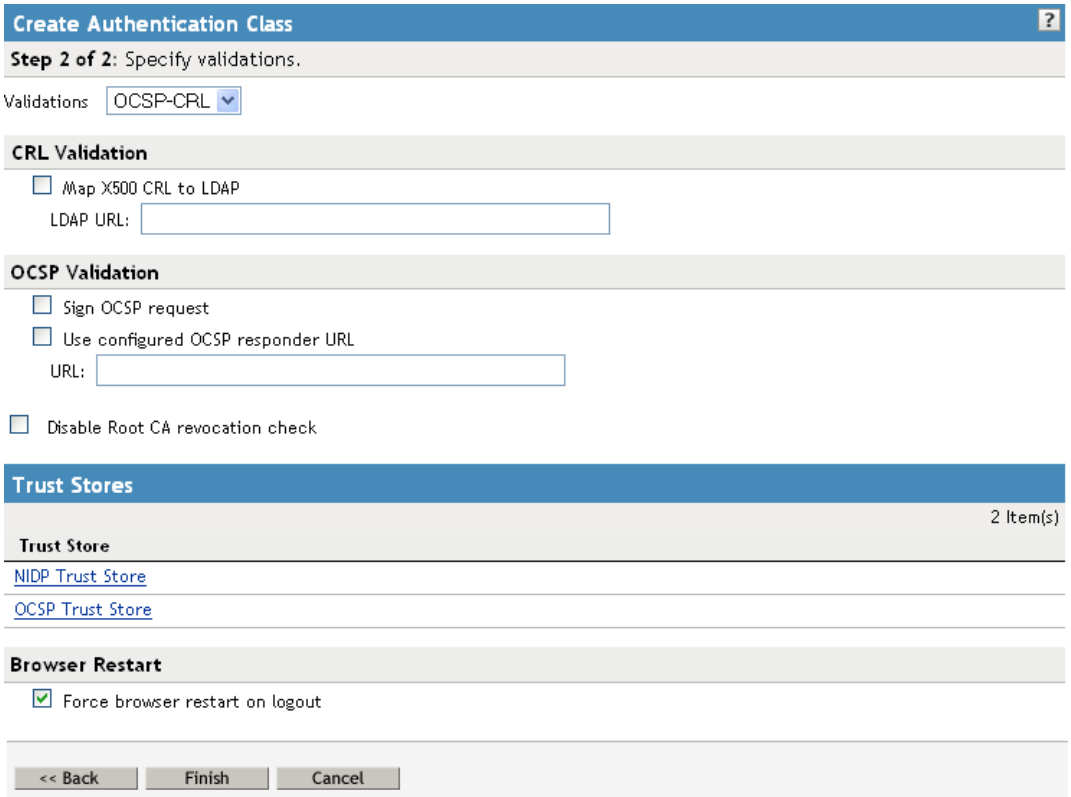
3 To create the X.509 authentication class, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Classes*.

4 Click *New*.



5 Specify a display name, then select *X509Class* from the drop-down menu.

6 Click *Next*.



7 Configure the validation options:

Validations: The validation type. Trust validation occurs if the certificate chain is verified in the *NIDP Trust Store*. In addition to usual certificate validations, the Identity Server supports CRL (certificate revocation list) and OCSP (Online Certificate Status Protocol) validations for each authentication request.

Access Manager caches CRLs, so the revoked status of a newly revoked certificate is not picked up until the next cache refresh. For higher security requirements, use OCSP validation with CRL validation. You can select None, CRL, OCSP, OCSP-CRL, or CRL-OCSP validation. In a production environment, for highest security, select either OCSP-CRL or CRL-OCSP validation. The default setting is to check OCSP first, then CRL.

CRL Validation: Checks the CRL. If you enable CRL validations, the CRL distribution point extension is read out of the user's X.509 certificate. The CRL distribution point contains the URL where the complete CRL can be found, as published by the certificate authority. The system checks the CRL itself and then checks to see if the user certificate is in the revoked list. The system can get the CRL over HTTP and LDAP. If you are not expecting the distribution point in user certificates, you can specify a value in the *LDAP URL* option to get the CRL.

Access Manager supports two schemes for a URL: `http://` and `ldap://`.

OCSP Validation: If OCSP validation is enabled, the Authority Info Access point (AIA) is read out of the user certificate, which contains the URL for the OCSP responder. A signed OCSP request for the user certificate is sent to OCSP responder. A signed OCSP response is received from the responder that has the revoked status for the user certificate. Alternately, if you are not expecting an AIA in a user certificate, you can specify a value in the OCSP responder *URL* field. The value you enter here overrides any OCSP responder URLs in a certificate.

Access Manager supports two schemes for a URL: `http://` and `ldap://`.

Disable Root CA Revocation Check: Disables whether to check if a certificate authority has been revoked. This option checks the CRL and OCSP for the trusted root certificate in the chain. You can enable or disable this option for X.509 user authentication performance.

If you enable the root CA revocation check, what the Identity Server checks depends upon the certificates that have been added to the Identity Server trust store. If the root certificate and the intermediate certificates in the chain are in the trust store, the Identity Server only validates the client (leaf) certificate. If the trust store only contains the root certificate, the browser sends the intermediate and leaf certificates, which are then validated by the Identity Server.

8 Configure the trust stores:

NIDP Trust Store: This trust store must contain the trusted root certificate of the certificate authorities that signed your user certificates. Click this link to add certificates to the trust store.

OCSP Trust Store: This trust store must contain the signing certificate of the OCSP servers you want to trust. Click this link to add certificates to the trust store. You must add the signing certificate, not the trusted root certificate, for this feature to work.

9 Configure the browser restart option.

Some browsers, such as Internet Explorer, keep the SSL session active until the user closes the browser. When the user logs in with the certificate on a smart card, then removes the card and logs out but does not close the browser, the SSL session is still active. If another user has access to the machine, that user can use the existing session.

To prevent this from happening, enable the *Force browser restart on logout* option.

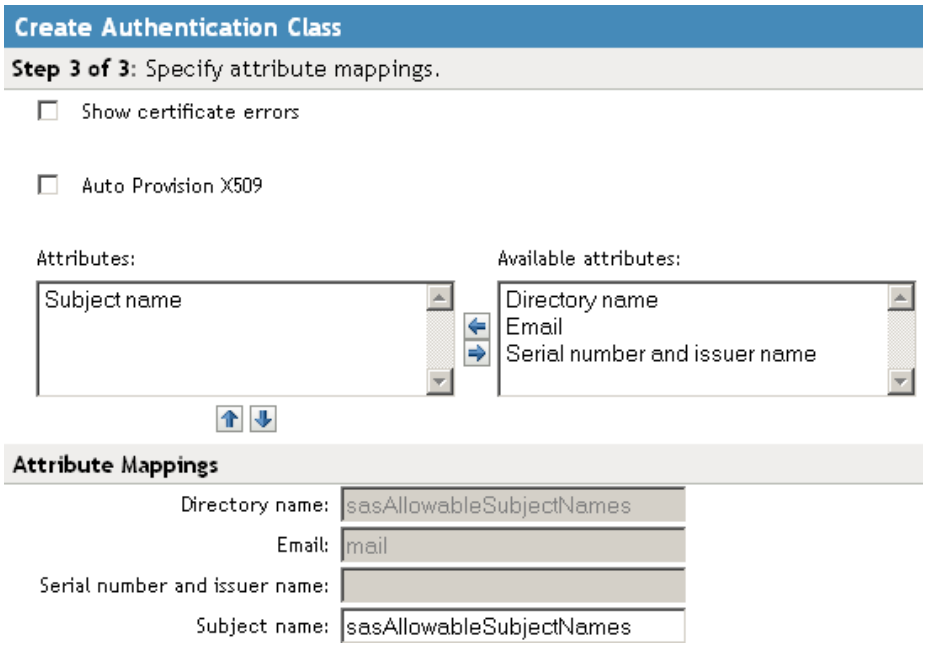
10 Click *Next*.

11 Continue with Section 4.2.1, "Configuring Attribute Mappings," on page 145.

4.2.1 Configuring Attribute Mappings

The attribute mapping options allow you to specify how the Identity Server maps the certificate to a user in the user store. *Subject name* is the default map.

1. Step 3 of the wizard or click *Devices* > *Identity Servers* > *Edit* > *Local* > *Classes* > *[Name of X.509 class]* > *Properties* > *Attributes*.
2. Configure attribute mappings.



Show certificate errors: Displays an error page when a certificate error occurs. This option is disabled by default.

Auto Provision X509: Enables using X.509 authentication for automatic provisioning of users. This option allows you to activate X.509 for increased security, while using a less secure way of authentication, such as username/password. Extra security measures can even include manual intervention to activate X.509 authentication by adding an extra attribute that is checked during authentication.

An example of using this option is when a user authenticates with an X.509 certificate, a lookup is performed for a matching SASallowableSubjectNames with the name of the user certificate. When no match is found, and *Auto Provision X509* is enabled, the user is presented with a custom error page specifying to click a button to provide additional credentials, such as a username and password, or to start an optional Identity Manager workflow. If the authentication is successful, then the user's SASallowableSubjectNames attribute is filled in with the certificate name of the user certificate.

When *Auto Provision X509* is enabled, and the attribute that is used for subject name mapping is changed from the default sasAllowableSubjectNames, you need to ensure that the LDAP attribute that is used can store string values with a length as long as the longest client certificate subject name. For example, if you use the LDAP attribute title (which has an upper bound of 64 characters) the *Auto Provision X509* fails the provisioning part of the authentication if the client certificate subject name is longer 64 characters. The authentication works if a valid name and password is given. However, provisioning fails.

Attributes: The list of attributes currently used for matching. If multiple attributes are specified, the evaluation of these attributes should resolve to only one user in the user store.

The evaluation first does a DN lookup for subject name or directory name mapping. If this fails, the rest of the mappings are looked up in a single LDAP query.

Available attributes: The available X.509 attributes. To use an attribute, select it and move it to the *Attributes* list. When the attribute is moved to the *Attributes* list, you can modify the mapping name in the *Attribute Mappings* section. The mapped name must match an attribute in your LDAP user store.

Directory name: Searches for the directory address in the client certificate and tries to match it to the DN of a user in the user store. If that fails, it searches the sasAllowableSubjectNames attribute of all users for a value that matches. The sasAllowableSubjectNames attribute must contain values that are comma-delimited, with a space after the comma. (For example, O=CURLY, OU=Organization CA or OU=Organization CA, O=CURLY.)

Email: Searches for the email attribute in the client certificate and tries to match it with a value in the LDAP mail attribute.

Serial number and issuer name: Lets you match a user's certificate by using the serial number and issuer name. The issuer name and the serial number must be put into the same LDAP attribute of the user, and the name of this attribute must be listed in the *Attribute Mappings* section.

When using a Case Ignore String attribute, both the issuer name and the serial number must be in the same attribute separated by a dollar sign ($) character. The issuer name must precede the $ character, with the serial number following the $ character. Do not use any spaces preceding or following the $ character. For example: O=CURLY, OU=Organization CA$21C0562C5C4

The issuer name can be from root to leaf or from leaf to root. The issuer name must be comma-delimited with a space after the comma. (For example, O=CURLY, OU=Organization CA or OU=Organization CA, O=CURLY.)

The serial number cannot begin with a zero (0) or with a hexadecimal notation (0x). If the serial number is 0x0BAC05, the value of the serial number in the attribute must be BAC05. The certificate number is displayed in Internet Explorer with a space after every fourth digit. However, you should enter the certificate number without using spaces.

The LDAP attribute can be any Case Ignore List or Case Ignore String attribute of the user. If you are configuring your own attribute, ensure that the attribute is added to the Person class. When using a Case Ignore List attribute, both the issuer name and the serial number must be in the same list. The issuer name needs to be the first item in the list, with the serial number being the second and last item in the list.

Subject name: Searches for the Subject name of the client certificate and tries to match it to the DN of a user in the user store. If that fails, it searches the sasAllowableSubjectNames attribute of all users for a value that matches the Subject name of the client certificate. The sasAllowableSubjectNames attribute must contain values that are comma-delimited, with a space after the comma. (For example, O=CURLY, OU=Organization CA or OU=Organization CA, O=CURLY.)

3 Click *Finish*.

4 Create a method for this class.

For instructions, see Section 3.3, "Configuring Authentication Methods," on page 123.

5 Create a contract for the method:

For instructions, see Section 3.4, "Configuring Authentication Contracts," on page 125.

If you want the user's credentials available for Identity Injection policies, add the password fetch method as a second method to the contract. For more information about this class and method, see Section 4.5, "Configuring Password Retrieval," on page 150.

6 Update the Identity Server.

4.2.2 Setting Up Mutual SSL Authentication

SSL provides the following security services from the client to the server:

- Authentication and nonrepudiation of the server, using digital signatures
- Data confidentiality through the use of encryption
- Data integrity through the use of authentication codes

Mutual SSL provides the same things from the server to the client as SSL. It provides authentication and nonrepudiation of the client, using digital signatures.

1 Set up Access Manager certificates for security, and import them into the Access Manager system. (See "Creating Certificates" in the *Novell Access Manager 3.1 SP2 Administration Console Guide*.)

2 Create an X.509 authentication class. (See Section 4.2, "Configuring Mutual SSL (X.509) Authentication," on page 142.)

3 Create an authentication method using this class. (See Section 3.3, "Configuring Authentication Methods," on page 123.)

4 Create an authentication contract using the X.509 method. (See Section 3.4, "Configuring Authentication Contracts," on page 125.)

5 Update the Identity Server cluster configuration. (See Section 14.1.1, "Updating an Identity Server Configuration," on page 322.)

6 Update any associated Access Gateways to read the new authentication contract.

7 Assign the contract to protect resources.

See "Configuring Protected Resources" in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*.

8 Update the Access Gateway.

4.3 Creating an ORed Credential Class

Access Manager includes a class that can be configured to accept any combination of name/password, X.509, or RADIUS credentials. When this class executes as part of a contract, users can select and enter their preferred type of credential.

For example, if a name/password credential is ORed with an X.509 credential, the user can select to use a certificate or to enter a name and password. As an administrator, you have decided that both credentials are equally secure for the protected resource the contract is protecting.

To create an ORed credential class:

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Classes*.

2 Click *New*, then fill in the following fields:

Display name: Specify a name for the class.

Java class: Select NPOrRadiusOrX509Class.

3 Click *Next*, then select the types of classes you want to OR. You must select at least one of the following:

Use Name/Password: Select this option if you want the PasswordClass to be one of the authentication options available to the user.

Use Radius: Select this option if you want the RadiusClass to be one of the authentication options available to the user.

Use X509: Select this option if you want the X509Class to be one of the authentication options available to the user.

4 (Conditional) If you want to use the protected version of the PasswordClass or RadiusClass, select the *Enforce use of HTTPS* option.

5 (Conditional) If you selected the *Use Name/Password* option, configure the properties:

5a In the *Name/Password Properties* section, click *New*.

5b Specify a property name and property value.

For information about the properties that the PasswordClass and the ProtectedPasswordClass support, see Section 3.2.2, "Specifying Common Class Properties," on page 121.

5c Click *OK*.

5d Repeat Step 5a through Step 5c to add more than one property.

6 Click *Next*.

7 (Conditional) If you selected the *Use Radius* option, configure the Radius properties.

For information about the configuration options, see Section 4.1, "Configuring for RADIUS Authentication," on page 141.

8 (Conditional) If you selected the *Use X509* option, configure how the certificate is validated.

For information about the configuration options, see Section 4.2, "Configuring Mutual SSL (X.509) Authentication," on page 142.

9 Click *Next*.

10 (Conditional) If you selected the *Use X509* option, configure the attribute mappings.

For information about the configuration options, see Section 4.2, "Configuring Mutual SSL (X.509) Authentication," on page 142.

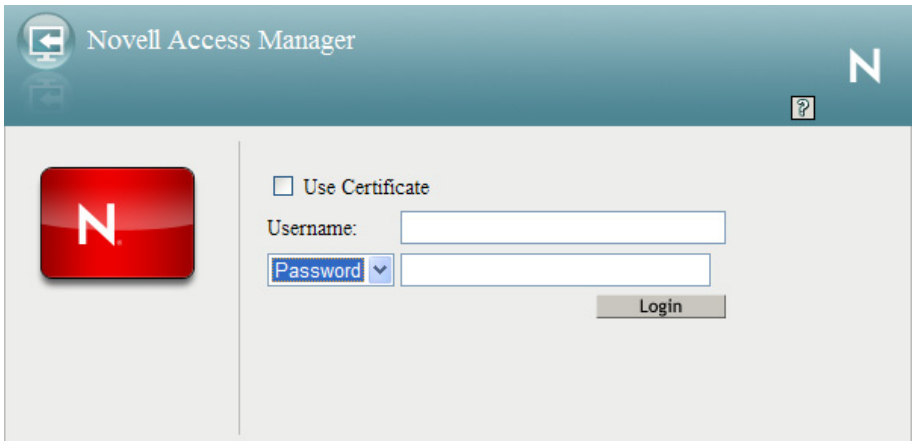
11 Click *Next*.

12 Click *Finish*.

13 Continue with creating a method and a contract for this class.

For configuration information, see Section 3.3, "Configuring Authentication Methods," on page 123 and Section 3.4, "Configuring Authentication Contracts," on page 125.

If the contract allows the user to select from the three types of credentials, the login page looks similar to the following:



The Radius class prompts the user for a token instead of a password. The user can use the drop-down menu to select between the password and the token. If the user selects to send a certificate, the username and password/token options become unavailable.

4.4 Configuring for OpenID Authentication

OpenID is an open, decentralized method for identifying users which allows users to use the same digital identity for logging in to multiple services. You can configure the Identity Server to trust the provider or providers of OpenIDs by configuring the OpenID class. You then configure a method and contract and assign a protected resources to use the contract for authentication. When the users supply the OpenID, they are granted access if the Identity Server has been configured to trust the provider of the OpenID server.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Classes*.
2. Click *New*, then fill in the following fields:

   Display name: Specify a name for the class.

   Java class: Select OpenIdClass.

   The Java class path is configured automatically.
3. Click *Next*, then configure the following properties:

   Open ID Provider Substrings: Specify at least one URL substring of an OpenID provider. The OpenID URL that user enters during the login process must contain one of the strings as a subset of the OpenID URL. For example, if user enters `https://user123.myopenid.com`, this field needs to contain one of the following strings:

   ```
   myopenid.com
   .myopenid.com
   ```

   To specify multiple URLs, separate them with a semicolon (;)

Identity the OpenID user locally: After the user authenticates at the OpenID provider, Access Manager can associate a username from the user store with the OpenID user. With this association, Access Manager can use the policies defined for the username to enforce access to protected resources.

- When this option is not selected, the OpenID user is not mapped to a local user. The username of the authenticated user remains as the OpenID URL. For example, if the user enters http://user123.myopenid.com for the URL, `http://user123.myopenid.com` becomes the username.
- When this option is selected, an attempt is made to map the OpenID user with a username in the user store. You can do this manually by storing the user's OpenID in the attribute specified in the *LDAP Attribute Name* option. You can also have the Identity Server add the OpenID value to the attribute by selecting the *Auto Provision LDAP Attribute* option.

LDAP Attribute Name: Specify the name of the attribute that contains the identification information for the users. For OpenID authentication, this attribute should contain the OpenID for the user.

Auto Provision LDAP Attribute: Select this option when you want the user to provide additional information for identification for the first authentication, such as a username and password. The Identity Server uses this information to identify the user, then writes the user's OpenID value to the attribute specified in the *LDAP Attribute Name* option. On subsequent logins, the Identity Server can identify the user by using the specified attribute and the user is not prompted for additional information.

4 Click *Finish*.

5 Create a method for this class.

For instructions, see Section 3.3, "Configuring Authentication Methods," on page 123.

6 Create a contract for the method:

For instructions, see Section 3.4, "Configuring Authentication Contracts," on page 125.

If you want the user's credentials available for Identity Injection policies, add the password fetch method as a second method to the contract. For more information about this class and method, see Section 4.5, "Configuring Password Retrieval," on page 150.

7 Update the Identity Server.

4.5 Configuring Password Retrieval

If you have configured contracts that do not use a username and password for the credentials and you want to configure single sign-on to protected resources that require a user's name and password, you need to configure the PasswordFetchClass to retrieve the user's name and password. You need to create the class, then create a method from the class. The method needs to be assigned as the second method for the authentication contract that does not prompt the user for a username and password. When the Identity Server executes the contract, the PasswordFetchClass retrieves the username and password and stores them with the LDAP credentials, which makes them available for Identity Injection policies.

IMPORTANT: The PasswordFetchClass only works with eDirectory user stores.

---

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Classes*.

2 Click *New*, then fill in the following fields:

Display name: Specify a name for the class.

Java class: Select *PasswordFetchClass*.

The Java class path is configured automatically.

3 Click *Next*, then configure the following general properties:

Ignore password retrieval failure: Select this option if you want users to continue with their sessions when the Identity Server can't retrieve their passwords. If this option is not selected, users are denied access when their passwords can't be retrieved.

Password to be retrieved: If your users have been configured to use a universal password, select *Universal Password*. Otherwise, select *Simple Password*.

NOTE: Universal Password Retrieval options needs to be properly set in the configuration of the Universal Password policy, so that it allows the password to be retrieved from the User Store.

For more information on Unable to retrieve Universal Password from eDirectory using PasswordFetchClass issue, see TID 7007114 ((https://kcs.innerweb.novell.com/contactcenter/php/viewdocument.do?externalId=7007114))

4 Click *Finish*.

5 Create a method for this class.

For instructions, see Section 3.3, "Configuring Authentication Methods," on page 123.

6 Assign the password fetch method as the second method for a contract that is using one of the following for its authentication method:

- RADIUS. See "Configuring for RADIUS Authentication" on page 141.
- X.509. See "Configuring Mutual SSL (X.509) Authentication" on page 142.
- OpenID. See "Configuring for OpenID Authentication" on page 149.
- Smart Card. See "Configuring Access Manager for NESCM" on page 153.
- Kerberos. See "Configuring for Kerberos Authentication" on page 163.

7 Update the Identity Server.

4.5.1 Retrieving Password from Different User Stores

The PasswordFetchClass has been enhanced to retrieve passwords from different user stores, than the authenticated user store by configuring properties of the authentication method in 3.1 SP2 IR3.

NOTE: Make sure that you have appropriate policies and permissions configured in the eDirectory to retrieve passwords.

4.5.2 Password Retrieval for Different User Store Lookup Settings

The Novell Access Manager supports password retrieval of the users who are mapped in the following ways:

1. CN to CN
2. distinguishedName (DN) to LDAP Attribute

NOTE: Please do not edit any property values and use the same values as mentioned in the examples below.

CN to CN Mapping

The CN users are mapped between two different user stores.

For Example - Active Directory CN is mapped with eDirectory CN for retrieving the password from eDirectory user store.

To achieve the password retrieval for this case add the below properties to Authentication Method which is using Passwordfetchclass

```
Property name  = com.novell.nidp.authentication.local.pwdfetch.userStoreToUse
Property value= local
Property name = com.novell.nidp.authentication.local.pwdfetch.userLookupType
Property value= usingCN
```

If the property value of the `com.novell.nidp.authentication.local.pwdfetch.userStoreToUse` is equal to `local`, then the passwordfetchclass tries fetching the password from the current user store (eDirectory). The principal user store is where the user is already authenticated using previous method (Active Directory).

If the property value of the com.novell.nidp.authentication.local.pwdfetch.userLookupType is equal to usingCN then the passwordfetchclass tries fetching the password from the current user store (eDirectory) by mapping the CN users between two different user stores.

distinguishedName to LDAP Attribute

The user names are detected and handled in LDAP attribute or DN users of theActive Directory are mapped with LDAP attribute of the eDirectory.

For Example - Active Directory DN is mapped with eDirectory LDAP attribute `samAccountName` for retrieving the password from eDirectory user store.

To achieve the password retrieval for this case add the below properties to Authentication Method, which is using Passwordfetchclass.

```
property name  = com.novell.nidp.authentication.local.pwdfetch.userStoreToUse
property  value= local
property name = com.novell.nidp.authentication.local.pwdfetch.userLookupType
property  value= usingAttr
property name = com.novell.nidp.authentication.local.pwdfetch.attributeName
property  value = ldapattributename
property name =
com.novell.nidp.authentication.local.pwdfetch.attributeAutoProvision
property value = false
```

If the property value of the `com.novell.nidp.authentication.local.pwdfetch.userStoreToUse` is equal to `local`, then the passwordfetchclass tries fetching the password from the current user store (eDirectory).The principal user store is where the user is already authenticated using previous method (Active Directory).

If the property value of the com.novell.nidp.authentication.local.pwdfetch.userLookupType is equal to usingAttr,then the passwordfetchclass tries fetching the password from the current user store (eDirectory) by mapping the DN users of the Active Directory with LDAP attribute of the eDirectory.

If the property value of the `com.novell.nidp.authentication.local.pwdfetch.attributeName` is equal to `ldapattributename` (`samAccountName`), then the passwordfetchclass tries fetching the password from the current user store based on the value of the LDAP attribute `samAccountName`, which are mapped to DN users of the Active Directory.

If the property value of the `com.novell.nidp.authentication.local.pwdfetch.attributeAutoProvision` is equal to `false`, then the passwordfetchclass tries fetching the password from LDAP attribute (samAccountName) which has the value of the DN users of the Active Directory and retrieves the password.

If the property value of the `com.novell.nidp.authentication.local.pwdfetch.attributeAutoProvision` is equal to `true`, then the passwordfetchclass tries fetching the password from LDAP attribute (samAccountName) which has the value of the DN users of the Active Directory and retrieves the password, else it prompts to log in to the eDirectory.

If the log in is successful, then the LDAP attribute (`samAccountName`) value populates in the DN user of the Active Directory. Next time when the user is logged in the same value is used.

4.6 Configuring Access Manager for NESCM

To use a smart card with Access Manager, you need to configure Access Manager to use the eDirectory server where you have installed the Novell Enhanced Smart Card Login Method for NMAS (NESCM). You then need to create a contract that knows how to prompt the user for the smart card credentials. The last task is to assign this contract to the protected resources that you want protected with a smart card. The following sections describe the prerequisites and the tasks:

- Section 4.6.1, “Prerequisites,” on page 154
- Section 4.6.2, “Creating a User Store,” on page 154
- Section 4.6.3, “Creating a Contract for the Smart Card,” on page 156

- Section 4.6.4, “Assigning the NESCM Contract to a Protected Resource,” on page 160
- Section 4.6.5, “Verifying the User’s Experience,” on page 160
- Section 4.6.6, “Troubleshooting,” on page 161

4.6.1 Prerequisites

- ❑ Make sure you can authenticate to the eDirectory server by using the smart card from a workstation.
  - The NESCM method needs to be installed on the eDirectory server and the workstation. See “Installing the Method” (http://www.novell.com/documentation/iasclient30x/nescm_install/data/b7gx5la.html) in the *Novell Enhanced Smart Card Method Installation and Administration Guide* (http://www.novell.com/documentation/iasclient30x/nescm_install/data/bookinfo.html).
  - The NESCM method needs to be configured. See “Configuring the Server” (http://www.novell.com/documentation/iasclient30x/nescm_install/data/b7tf2gi.html) in the *Novell Enhanced Smart Card Method Installation and Administration Guide* (http://www.novell.com/documentation/iasclient30x/nescm_install/data/bookinfo.html).
  - Provision your smart card according to your company policy.
- ❑ Make sure you have a basic Access Gateway configuration with a protected resource that you want to protect with a smart card. For more information, see the *Novell Access Manager 3.1 SP2 Installation Guide* and the *Novell Access Manager 3.1 SP2 Setup Guide*.

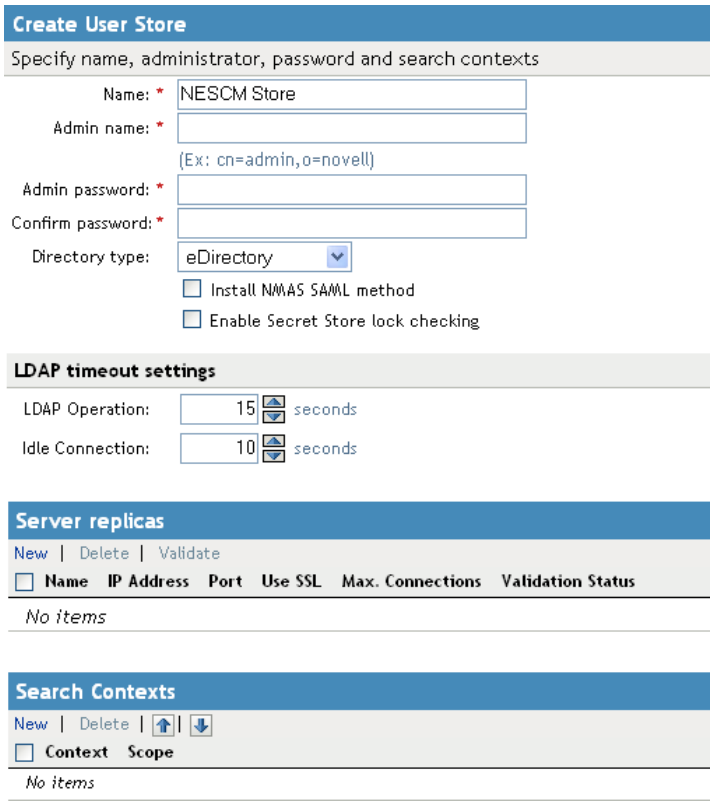
4.6.2 Creating a User Store

The Identity Server must be configured to use the eDirectory replica where you have installed the NESCM server method.

- If you have already configured the Identity Server to use this replica, skip this section and continue with Section 4.6.3, “Creating a Contract for the Smart Card,” on page 156.
- If your Identity Server is using a different user store, you need to configure the Identity Server.

To configure the Identity Server for the eDirectory replica that has the NESCM method:

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local*> *User Stores* > *New*.



2 On the *Create User Store* page, fill the following fields:

Name: A display name for the eDirectory replica (for example, `nescm_replica`).

Admin Name: The distinguished name of the admin user of the directory. Administrator-level rights are required for setting up a user store.

Admin Password and Confirm Password: The password for the admin user and the confirmation for the password.

Directory Type: Select eDirectory.

3 In the *Server replica* section, click *New*, and fill the following fields:

Name: The display name for the LDAP directory server (for example, `nescm_server`).

IP Address: The IP address of the LDAP directory server. The port is set automatically to the standard LDAP ports.

4 Click *Use secure LDAP connections*. You must enable SSL between the user store and the Identity Server. The port changes to 636, which is the secure LDAP port.

5 Click *Auto import trusted root*.

6 Click *OK* to confirm the import.

7 Select the *Root CA Certificate* to trust any certificate signed by that certificate authority.

8 Specify an alias, then click *OK*.

An alias is a name you use to identify the certificate used by Access Manager.

9 Click *Close*, then click *OK*.

10 Under *Server Replicas*, verify the *Validation Status*.

The system displays a green check mark if the connection is valid.

11 Set up a search context.

12 Click *Finish* to save the information.

13 Continue with Section 4.6.3, “Creating a Contract for the Smart Card,” on page 156.

4.6.3 Creating a Contract for the Smart Card

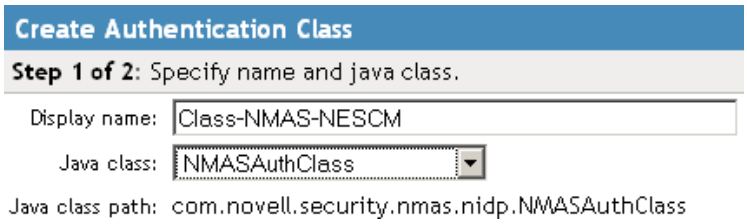
You need to create a contract that uses the NESCM method. To do this, you need to first create an NMAS class, then a method that uses that class. The last task is to create a contract that uses the method. The following sections describe these tasks:

- “Creating an NMAS Class for NESCM” on page 156
- “Creating a Method to Use the NMAS Class” on page 157
- “Creating an Authentication Contract to Use the Method” on page 158

Creating an NMAS Class for NESCM

When you create a class, you can specify values for properties. In the following steps, you specify a property value that determines the sequence of login prompts that the user receives when authenticating with a smart card.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Classes* > *New*.

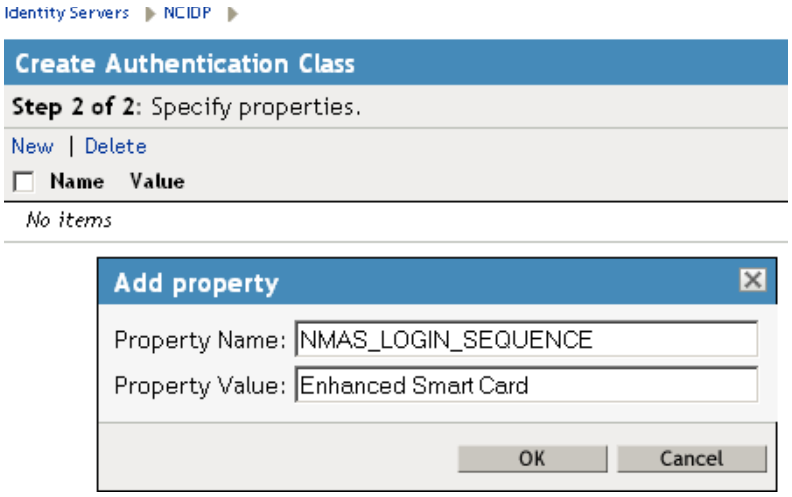


2 Specify a display name for the class (for example, `Class-NMAS-NESCM`).

3 For the *Java class*, select *NMASAuthClass* from the selection list.

4 Click *Next*.

5 On the *Specify Properties* page, click *New*.



6 Specify the following values for the property:

Property Name: Specify `NMAS_LOGIN_SEQUENCE`

Property Value: Specify `Enhanced Smart Card`

The Property Value matches the method name as displayed in the *NMAS* task > *NMAS Login Methods*.

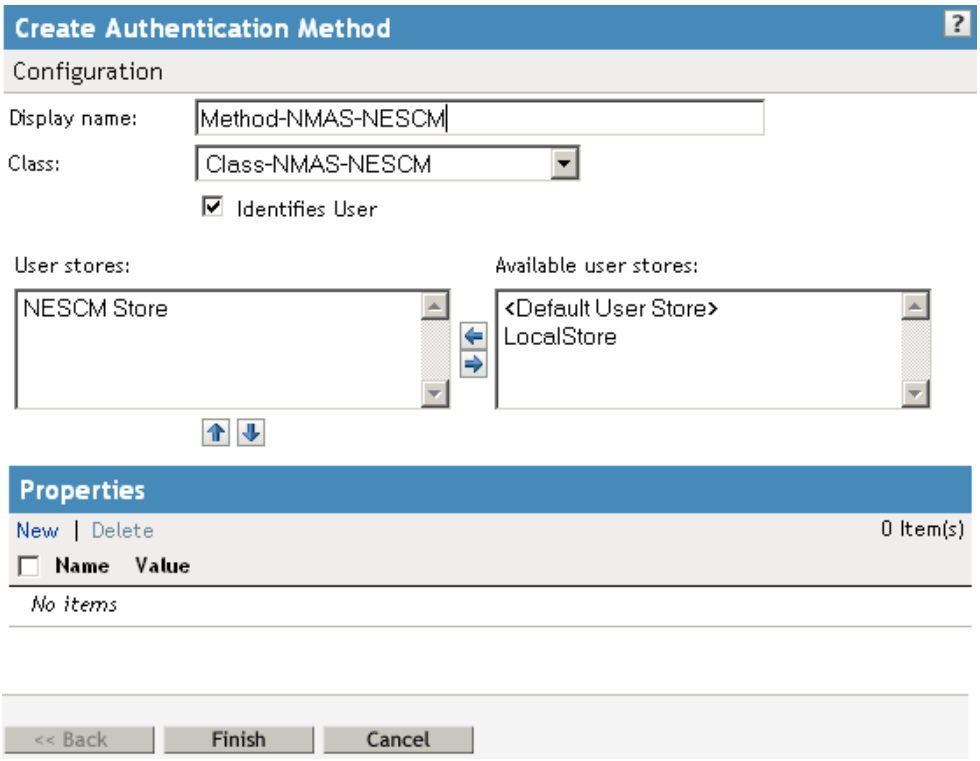
7 Click *OK*, then click *Finish*.

8 Continue with "Creating a Method to Use the NMAS Class" on page 157.

Creating a Method to Use the NMAS Class

When you create a method, you can specify property values that are applied to just this method and not the entire class. In this tutorial, we want the method to use the same login sequence as the class. The method also allows you to specify which user stores can use the method. For a smart card method, you need to ensure that the user store or stores specified for the method have NESCM installed.

1 On the Local page for the Identity Server, click *Methods* > *New*.



2 Specify a *Display name* (for example, `Method-NMAS-NESCM`).

3 From the *Class* selection list, select the class created in "Creating an NMAS Class for NESCM" on page 156.

4 In the *Available user stores list*, select the user store created in Section 4.6.2, "Creating a User Store," on page 154, then click the left-arrow to move this user store into the *User stores* list.

Leave other settings on this page unchanged.

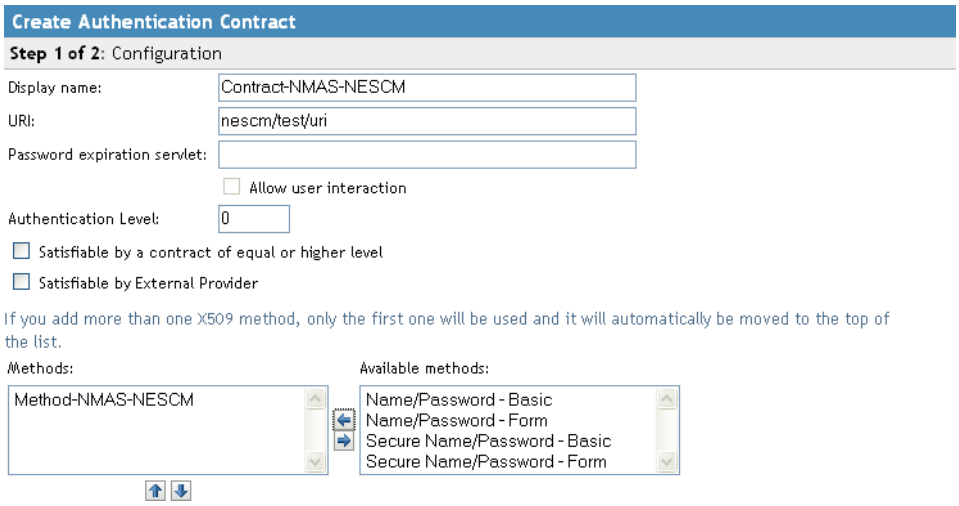
5 Click *Finish*.

6 Continue with "Creating an Authentication Contract to Use the Method" on page 158.

Creating an Authentication Contract to Use the Method

Contracts are the element you can assign to a protect a resource.

1 On the Local page for the Identity Server, click *Contracts* > *New*.



2 Specify a *Display name* (for example, `Contract-NMAS-NESCM-UserStore1`).

3 Enter a *URI* (for example, `nescm/test/uri`).

   The URI is used to identify this contract for external providers and is a unique path value that you create.

4 In the *Available methods* list, select the method created in "Creating a Method to Use the NMAS Class" on page 157, then click the left-arrow to move this method into the *Methods* list.

   All other fields can remain in the default state.

5 (Conditional) If you want the user's credentials (username and password) to be available for Identity Injection policies, add the password fetch method as a second method for the contract.

   For more information about this method and class, see Section 4.5, "Configuring Password Retrieval," on page 150.

6 Click *Next*, then configure a card for the contract by filling in the following fields:

   ID: (Optional) Specify an alphanumeric value that identifies the card. If you need to reference this card outside of the Administration Console, you need to specify a value here. If you do not assign a value, the Identity Server creates one for its internal use.

   Text: Specify the text that is displayed on the card to the user, for example Smart Card.

   Image: Select the image to display on the card. You can select the NMAS Biometrics image or you can select the *Select local image* option and upload an image that your users can associate with using this smart card authentication contract.

   Show Card: Determine whether the card is shown to the user, which allows the user to select and use the card for authentication. If this option is not selected, the card is only used when a service provider makes a request for the card.

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

8 Update the Identity Server.

9 Update the Access Gateway.

10 Continue with Section 4.6.4, "Assigning the NESCM Contract to a Protected Resource," on page 160

4.6.4 Assigning the NESCM Contract to a Protected Resource

Contracts must be created before they can be assigned to protected resources. The following steps explain how to assign the NESCM contract to an existing protected resource. If you have not created a protected resource, see "Configuring Protected Resources"in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*.

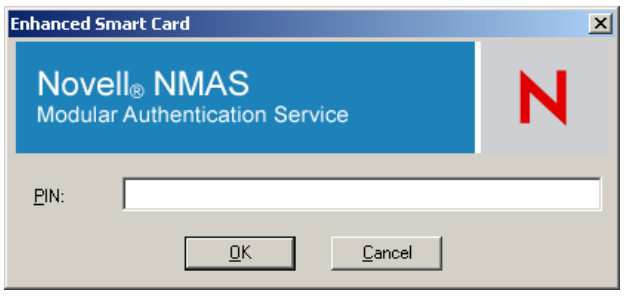
1. In the Administration Console, click *Devices* > *Access Gateways* > *Edit* > *[Name of Reverse Proxy]*.

   The reverse proxy should be configured with a resource that you want to protect with the smart card.
2. Click the *Protected Resource* link for the proxy service where you want to assign the NESCM contract.
3. To enable the NESCM contract on an existing protected resource, click the *Authentication Procedure* link for that resource, then select the NESCM contract created in "Creating an Authentication Contract to Use the Method" on page 158.

   If the contract is not listed, make sure you have updated the changes to the servers, first to the Identity Server and then the Access Gateway. If you have multiple Identity Server configurations, make sure that the Access Gateway is assigned to the Identity Server configuration that contains the NESCM contract (click *Access Gateways* > *Edit* > *Reverse Proxy / Authentication*).
4. Click *OK*.
5. Click the *Access Gateways* task, then update the Access Gateway.
6. Continue with Section 4.6.5, "Verifying the User's Experience," on page 160.

4.6.5 Verifying the User's Experience

1. From the smart-card-equipped workstation, browse to and select the URL of the proxy service where the protected resource requiring NESCM type authentication is enabled.
2. When prompted by Access Manager, enter a *username*.
3. When prompted for the smart card password, enter a password (the smart card PIN).



If the Smart Card contains a certificate that meets the defined criteria (in this example, a matching Subject name and trusted signing CA), the user is now successfully authenticated to the IDP and is connected through the Access Gateway to the protected resource.

4.6.6 Troubleshooting

Error	Resolution
Authentication fails without prompting the user for the token	Verify that you have configured the class and method correctly. See “Creating an NMAS Class for NESCM” on page 156 and “Creating a Method to Use the NMAS Class” on page 157.
Certificate validation fails	Verify that a trusted root object created for the signing CA of the certificate on the smart card exists in the eDirectory trusted root container.

Configuring for Kerberos Authentication

5

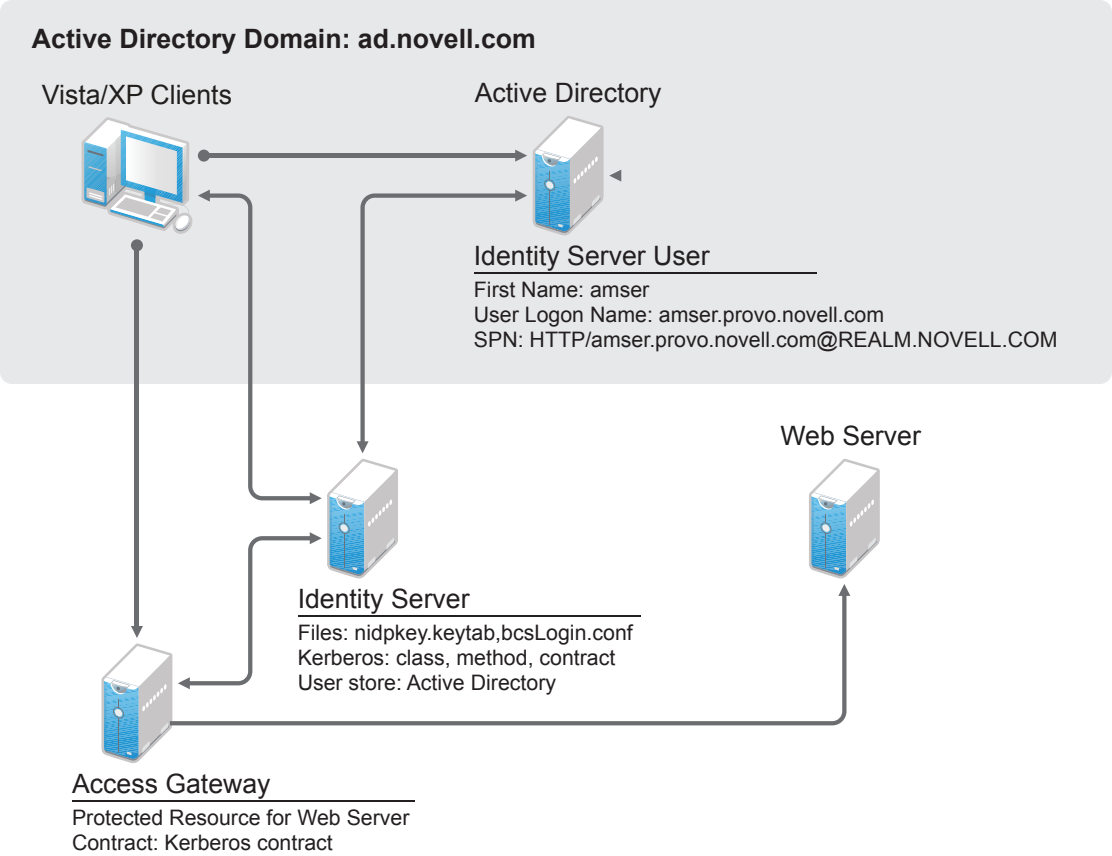
Kerberos is an authentication method that allows users to log in to an Active Directory domain. This authentication method provides them with a token, which an Identity Server can be configured to use as a contract. This provides single sign-on for the user between Active Directory and the Identity Server.

Kerberos authentication is achieved using SPNEGO with GSS-API (JGSS). SPNEGO (RFC 2478 - Simple and Protected GSSAPI Negotiation implementation in Microsoft Windows 2000/XP/2k3/2k8) is a GSSAPI mechanism for extending a Kerberos single-sign-on environment to Web transactions and services. It lets peers determine which GSSAPI mechanisms are shared and lets them select one and establish a security context with it. SPNEGO's most visible use is in Microsoft's HTTP Negotiate authentication mechanism.

The Kerberos module for Access Manager is implemented as additional out-of-the-box authentication mechanism to securely negotiate and authenticate HTTP requests for protected resources. This makes it possible to seamlessly authenticate (single-sign-on) to the Identity Server from enterprise-wide Microsoft Windows Domain Logon.

This section explains how to configure Active Directory, the Identity Server, and the Access Gateway for Kerberos authentication to a protected Web server. Figure 5-1 illustrates this configuration.

Figure 5-1 *Example Kerberos Configuration*



Kerberos requires the following configuration tasks:

- Section 5.1, “Prerequisites,” on page 164
- Section 5.2, “Configuring Active Directory,” on page 165
- Section 5.3, “Configuring the Identity Server,” on page 168
- Section 5.4, “Configuring the Clients,” on page 173
- Section 5.5, “Configuring the Access Gateway for Kerberos Authentication,” on page 175

5.1 Prerequisites

Kerberos authentication is supported for the following configuration:

- Clients must be running one of the following operating systems:

  Windows XP with Internet Explorer 7 or 8. Some minimal testing has been done with Internet Explorer 6. To make Kerberos work with Internet Explorer 6, you need to enable integrated Windows authentication. For information on how to enable this feature, see “Authentication Uses NTLM instead of Kerberos” (http://technet.microsoft.com/en-us/library/cc779070.aspx).

  Windows Vista with the latest version of Internet Explorer.

Windows 7 with Internet Explorer 8. Be aware of the following issues:

- Internet Explorer needs to have the Internet Options configured to trust the URL of the Identity Server.
- The keytab file must be configured to trust more than DES encryption. If you created your keytab file for an earlier version of Access Manager where only DES was supported, you need to recreate the keytab file. For the new procedure, see Section 5.2.3, "Configuring the Keytab File," on page 167.

For more information on these issues, see TID 7006036 (http://www.novell.com/support/viewContent.do?externalId=7006036&sliceId=1).

- Active Directory must be configured to contain entries for both the users and their machines. Active Directory must be running on Windows Server 2003 Enterprise SP2 or Windows Server 2008 SP2 or higher.
- Active Directory and the Identity Server must be configured to use a Network Time Protocol server. If time is not synchronized, authentication fails.
- If a firewall separates the Active Directory Server from the Identity Server, the firewall needs to open ports TCP 88 and UDP 88 so that the Identity Server can communicate with the KDC on the Active Directory Server.
- The Identity Server can communicate with only one KDC identified by IP address in the configuration. This limitation is caused by the underlying Sun JGSS and limits the Identity Server so that it can support only one Kerberos class with one Kerberos method.

5.2 Configuring Active Directory

You must create a new user in Active Directory for the Identity Server, set up this user account to be a service principal, create a keytab file, and add the Identity Server to the Forward Lookup Zone. These tasks are described in the following sections:

- "Installing the spn and the ktpass Utilities for Windows Server 2003" on page 165
- "Creating and Configuring the User Account for the Identity Server" on page 166
- "Configuring the Keytab File" on page 167
- "Adding the Identity Server to the Forward Lookup Zone" on page 167

5.2.1 Installing the spn and the ktpass Utilities for Windows Server 2003

When you install Windows Server 2003 and Active Directory, the spn and ktpass utilities are not installed in a default installation. These utilities are installed in a default Windows Server 2008 installation.

You need the spn and ktpass utilities to configure the Identity Server for Kerberos authentication.

1. Insert the Windows 2003 CD into the CD drive.
2. To install the utilities, run `\SUPPORT\TOOLS\SUPTOOLS.MSI` on the CD.

   The utilities are installed in `C:\Program Files\Support Tools`.

5.2.2 Creating and Configuring the User Account for the Identity Server

1. In *Manage Your Server* on your Windows server, select the *Manage users and computers in Active Directory* option.
2. Select to create a new user.
3. Fill in the following fields:

   First name: Specify the hostname of the Identity Server. This is the username. For the example configuration, this is `amser`.

   User logon name: Specify `HTTP/<Identity_Server_Base_URL>`. For this example configuration, your Identity Server has a base URL of `amser.provo.novell.com`, and you would specify the following for the *User Logon Name*:

   ```
   HTTP/amser.provo.novell.com
   ```

   The realm is displayed next to the *User logon name*.

   User logon name (pre Windows 2000): Specify the hostname of the Identity Server. The default value must be modified. For the example configuration, this is `amser`.
4. Click *Next*, and configure the password and its options:

   Password: Specify a password for this user

   Confirm password: Enter the same password.

   User must change password at next logon: Deselect this option.

   Password never expires: Select this option.
5. Click *Next*, then click *Finish*.

   This creates the Identity Server user. You need to remember the values you assigned to this user for *First name* and *User logon name*.
6. To set the servicePrincipalName (spn) attribute on this user, open a command window and enter the following command:

   ```
   setspn -A HTTP/<userLogonName> <userName>
   ```

   For this configuration example, you would enter the following command:

   ```
   setspn -A HTTP/amser.provo.novell.com@REALM.NOVELL.COM amser
   ```

   This adds the servicePrincipalName attribute to the user specified with the value specified in the `-A` parameter.
7. (Optional) Verify that the user has the required servicePrincipalName attribute with a valid value. Enter the following command:

   ```
   setspn -L <userName>
   ```

   For this configuration example, you would enter the following command:

   ```
   setspn -L amser
   ```

5.2.3 Configuring the Keytab File

The keytab file contains the secret encryption key that is used to decrypt the Kerberos ticket. You need to generate the keytab file and copy it to the Identity Server.

1 On the Active Directory server, open a command window and enter a `ktpass` command with the following parameters:

`ktpass /out value /princ value /mapuser value /pass value`

The command parameters require the following values:

Parameter	Value	Description
/out	*<outputFilename>*	Specify a name for the file, with `.keytab` as the extension. For example: `nidpkey.keytab`
/princ	*<servicePrincipalName>* *@<KERBEROS_REALM>*	Specify the service principal name for the Identity Server, then @, followed by the Kerberos realm. The default value for the Kerberos realm is the Active Directory domain name in all capitals. The Kerberos realm value is case sensitive.
/mapuser	*<identityServerUser>@<AD_DOMAIN>*	Specify the username of the Identity Server user and the Active Directory domain to which the user belongs.
/pass	<userPassword>	Specify the password for this user.

For this configuration example, you would enter the following command to create a keytab file named `nidpkey`:

```
ktpass /out nidpkey.keytab /princ HTTP/amser.provo.novell.com@AD.
NOVELL.COM /mapuser amser@AD.NOVELL.COM /pass novell
```

2 Copy the keytab file to the Identity Server.

Copy the file to the default location on the Identity Server:

Linux: `/opt/novell/java/jre/lib/security`

Windows Server 2003: `C:\Program Files\Novell\jre\lib\security`

Windows Server 2008: `C:\Program Files (x86)\Novell\jre\lib\security`

3 If the cluster contains multiple Identity Servers, copy the keytab file to each member of the cluster.

5.2.4 Adding the Identity Server to the Forward Lookup Zone

1 In *Manage Your Server* on your Windows server, click *Manage this DNS server*.

2 Click *Forward Lookup Zone*.

3 Click the Active Directory domain.

4 In the right pane, right click, and select *New Host (A)*.

5 Fill in the following fields:

Name: Specify the hostname of the Identity Server.

IP Address: Specify the IP address of the Identity Server.

6 Click *Add Host*.

5.3 Configuring the Identity Server

You need to configure the Identity Server to use the Active Directory server as a user store, configure a Kerberos authentication class, method, and contract, create a configuration file, enable logging to verify the configuration, then restart Tomcat. These instructions assume that you have installed and configured an Identity Server cluster configuration. If you have not, see the *Novell Access Manager 3.1 SP2 Installation Guide* and the *Novell Access Manager 3.1 SP2 Setup Guide*.

This section covers the following tasks:

5.3.1 Enabling Logging for Kerberos Transactions

Enabling logging is not required, but it is highly recommended. If Kerberos authentication does not function after you have finished the configuration tasks, the first step in solving the problem is to look at the `catalina.out` (Linux) or the `stdout.log` (Windows) file.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Logging*.

2 Enable the *File Logging* and *Echo To Console* options.

3 In the *Component File Logger Levels* section, set *Application* to *debug*.

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

5.3.2 Configuring the Identity Server for Active Directory

You need to either configure your Identity Server to use Active Directory as a user store or verify your existing configuration for your Active Directory user store.

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

2 Click *Local*.

3 View your installed user stores.

If you have already configured your Identity Server to use the Active Directory server, click its name.

If you haven't configured a user store for the Active Directory server, click *New*.

4 For a new user store, fill in the following fields. For an existing Active Directory user store, verify the values.

Name: Specify a name of the user store for reference.

Admin name: Specify the name of the administrator of the Active Directory server. Administrator-level rights are required for setting up a user store. This ensures read/write access to all objects used by Access Manager.

Admin password and Confirm password: Specify the password for the administrator of the Active Directory server and confirm the password.

Directory Type: Select *Active Directory*.

Search Contexts: For a new user store, click *New* and specify the context of the administrator of the Active Directory server. For an existing user store, verify that you have an entry for the context of the administrator and add one if it is missing.

5 (Conditional) For a new Active Directory user store, add a replica. In the *Server replicas* section, click *New*.

5a Fill in the following fields:

Name: Specify a name of the replica for reference. This can be the name of your Active Directory server.

IP Address: Specify the IP address of the Active Directory server and the port you want the Identity Server to use when communicating with the Active Directory server.

5b Configure the other fields to fit your security model.

5c Click *OK*.

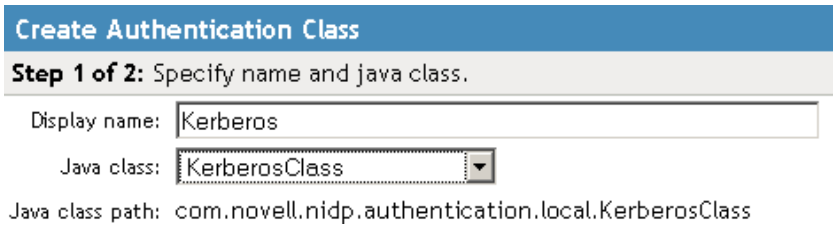
6 (Optional) Specify values for the other configuration options.

7 To save your changes, click *OK* or *Finish*.

8 Continue with "Creating the Authentication Class, Method, and Contract" on page 169.

5.3.3 Creating the Authentication Class, Method, and Contract

1 In the Local page, click *Classes* > *New*.



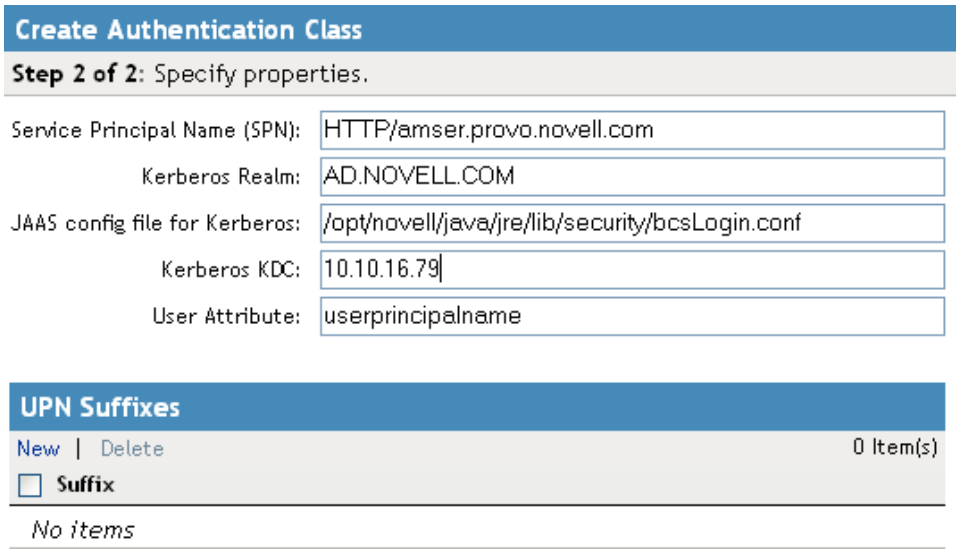
2 Fill in the following fields:

Display name: Specify a name that you can use to identify this class.

Java class: Select *KerberosClass*.

The *Java class path* field displays the name of the KerberosClass.

3 Click *Next.*



4 Fill in the following fields:

Service Principal Name (SPN): Specify the value of the servicePrincipalName attribute of the Identity Server user. For this example configuration, this is `HTTP/amser.provo.novell.com`.

Kerberos Realm: Specify the name of the Kerberos realm. The default value for this realm is the domain name of the Active Directory server, entered in all capitals. The value in this field is case sensitive. For this example configuration, this is `AD.NOVELL.COM`.

JAAS config file for Kerberos: Verify the default path. This should be the same path to which you copied the keytab file (see Step 2 in "Configuring the Keytab File" on page 167) and end with the name of the configuration file, `bcsLogin.conf`.

For Windows, the path needs to contain double slashes, for example: `C:\\Program Files\\Novell\\jre\\lib\\security`

Instructions for creating this file are in "Creating the bcsLogin Configuration File" on page 172.

Kerberos KDC: Specify the IP address of the Active Directory server.

User Attribute: Specify the name of the Active Directory attribute that combines the cn of the user with the DNS domain name to form its value. It is an alternate name for user login. Accept the default value unless you have set up a different attribute.

5 (Conditional) If you have configured your users to have multiple User Principal Names (UPN) so they can log in using different names (such as jdoe@abc.com, jdoe@bcd.com, and jdoe@cde.com), click *New*, specify the suffix (such as @abc.com), then click *OK*.

6 Click *Finish*.

IMPORTANT: You should create only one Kerberos class. This is caused by a limitation in the underlying Sun JGSS.

---

7 On the Local page, click *Methods* > *New*.

8 Fill in the following fields:

Display name: Specify a name that you can use to identify this method.

Class: Select the class that you created for Kerberos.

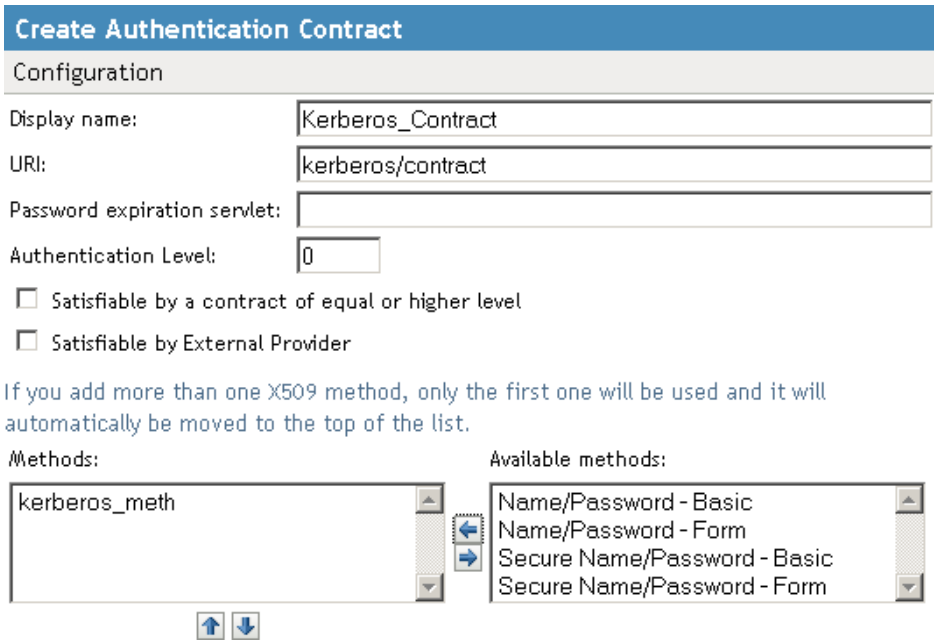
User stores: Move the Active Directory user store to the list of User stores. If you have only one installed user store, <*Default User Store*> can be used. If you have multiple user stores, the Active Directory user store must be in this list (or if it is configured to be the default user store, <*Default User Store*> must be in this list).

NOTE: The testing procedure to verify Kerberos authentication is dependent upon having the Active Directory user store configured as the default user store. See Step 13.

You do not need to configure properties for this method.

9 Click *Finish*.

10 In the Local page, click *Contracts* > *New*.



11 Fill in the following fields:

Display name: Specify a name that you can use to identify this method.

URI: Specify a value that uniquely identifies the contract from all other contracts.

The URI cannot begin with a slash, and it must uniquely identity the contract. For example: `kerberos/contract`

Methods: From the list of *Available methods*, move your Kerberos method to the *Methods* list.

You do not need to configure the other contract options.

12 Click *Finish*.

13 (Optional) To use the procedure that verifies the authentication configuration, you need to make the Active Directory user store the default user store. In the Local page, click *Defaults*.

13a Fill in the following fields:

User Store: Select the name of your Active Directory user store.

Authentication Contract: Select the name of your Kerberos contract.

13b Click *OK*.

This allows you to log in directly to the Identity Server by using the Kerberos contract. If you have already logged in to the Active Directory domain on the Windows machine, single sign-on is enabled and you are not prompted to log in to the Identity Server.

14 On the Identity Servers page, click *Update*.

Wait until the Health icon turns green. Click *Refresh* to update the page.

15 If you have Access Gateways or J2EE Agents that you want to configure to use the Kerberos contract, update these devices so that the Kerberos contract is available.

16 Continue with "Creating the bcsLogin Configuration File" on page 172.

5.3.4 Creating the bcsLogin Configuration File

If you are upgrading from 3.0.4 to 3.1 SP2, the syntax of the `bcsLogin.conf` file has changed. For details, see "Upgrading the SP4 Identity Servers" in the *Novell Access Manager 3.1 SP2 Installation Guide*.

To create the file:

1 Open a text editor.

2 Enter the following lines. The file cannot contain any white space, only end-of-line characters. Two lines (principal and keyTab) need to specify unique information for your configuration. The principal line needs to specify the service principle name for the Identity Server. The keyTab line needs to specify the location of the keytab file. The following file uses the values of the example configuration for the principal and keyTab lines. The keyTab and ticketCache lines use the default path for SUSE Linux Enterprise Server (SLES).

```
com.sun.security.jgss.accept {
com.sun.security.auth.module.Krb5LoginModule required
debug="true"
useTicketCache="true"
ticketCache="/opt/novell/java/jre/lib/security/spnegoTicket.cache"
doNotPrompt="true"
principal="HTTP/amser.provo.novell.com@AD.NOVELL.COM"
useKeyTab="true"
keyTab="/opt/novell/java/jre/lib/security/nidpkey.keytab"
storeKey="true";
};
```

For Windows, the path needs to contain double slashes: `C:\\Program Files\\Novell\\jre\\lib\\security`

Windows Server 2003: The path in the keyTab line should be `C:\\Program Files\\Novell\\jre\\lib\\security\\nidpkey.keytab`

The path in the ticketCache line should be `C:\\Program Files\\Novell\\jre\\lib\\security\\spnegoTicket.cache`

Windows Server 2008: The path in the keyTab line should be `C:\\Program Files (x86)\\Novell\\jre\\lib\\security\\nidpkey.keytab`

The path in the ticketCache line should be `C:\\Program Files (x86)\\Novell\\jre\\lib\\security\\spnegoTicket.cache`

3 Save this file with a name of `bcsLogin.conf`.

4 Copy this file to the location specified in the *JAAS config file for Kerberos* field of Step 4 in "Creating the Authentication Class, Method, and Contract" on page 169.

5 Make sure the file permissions are set correctly. They should be set to 644.

6 Restart Tomcat.

- **Linux Identity Server:** Enter the following command:

  ```
  /etc/init.d/novell-tomcat5 restart
  ```

- **Windows Identity Server:** Enter the following commands:

  ```
  net stop Tomcat5
  ```

  ```
  net start Tomcat5
  ```

Whenever you make changes to the `bcsLogin.conf` file, you need to restart Tomcat.

7 If the cluster contains multiple Identity Servers, copy the `bcsLogin.conf` file to each member of the cluster, then restart Tomcat on that member.

5.3.5 Verifying the Kerberos Configuration

To view the `catalina.out` (Linux) or the `stdout.log` (Windows) file of the Identity Server:

1 In the Administration Console, click *Auditing* > *General Logging*.

2 In the Identity Servers section, select the `catalina.out` or `stdout.log` file.

3 Download the file and open it in a text editor.

4 Search for Kerberos and verify that a subsequent line contains a `Commit Succeeded phrase`. For the configuration example, the lines look similar to the following:

```
principal's key obtained from the keytab
principal is HTTP/amser.provo.novell.com@AD.NOVELL.COM
Added server's keyKerberos Principal HTTP/
amser.provo.novell.com@AD.NOVELL.COMKey Version 3key EncryptionKey:
keyType=3 keyBytes (hex dump)=0000: CB 0E 91 FB 7A 4C 64 FE

[Krb5LoginModule] added Krb5Principal HTTP/
amser.provo.novell.com@AD.NOVELL.COM to Subject
Commit Succeeded
```

5 If the file does not contain any lines similar to these, verify that you have enabled logging. See "Enabling Logging for Kerberos Transactions" on page 168.

6 If the commit did not succeed, search backward in the file and verify the following values:

- Service Principal Name
- Name of keytab file

For the example configuration, the file should contain lines with text similar to the following:

```
Principal is HTTP/amser.provo.novell.com
KeyTab is /usr/lib/java/jre/lib/security/nidpkey.keytab
```

7 (Conditional) If you make any modifications to the configuration, either in the Administration Console or to the bcsLogin file, restart Tomcat on the Identity Server.

5.4 Configuring the Clients

1 Add the computers of the users to the Active Directory domain.

For instructions, see your Active Directory documentation.

2 Log in to the Active Directory domain, rather than the machine.

3 (Conditional) If you are using Internet Explorer, configure the Web browser to trust the Identity Server:

3a Click *Tools* > *Internet Options* > *Security* > *Local intranet* > *Sites* > *Advanced*.

3b In the *Add this website to the zone* text box, enter the Base URL for the Identity Server, then click *Add*.

In the configuration example, this is `http://amser.provo.novell.com`.

3c Click *Close* > *OK*.

3d Click *Tools* > *Internet Options* > *Advanced*.

3e In the Security section, select *Enable Integrated Windows Authentication*, then click *OK*.

3f Restart the browser.

4 (Conditional) If you are using Firefox, configure the Web browser to trust the Identity Server:

4a In the URL field, specify `about:config`.

4b In the *Filter* field, specify *network.n*.

4c Double click `network.negotiate-auth.trusted-uris`.

This preference lists the sites that are permitted to engage in SPNEGO Authentication with the browser. Specify a comma-delimited list of trusted domains or URLs.

For this example configuration, you would add `http://amser.provo.novell.com` to the list.

4d If the deployed SPNEGO solution is using the advanced Kerberos feature of Credential Delegation, double-click `network.negotiate-auth.delegation-uris`. This preference lists the sites for which the browser can delegate user authorization to the server. Specify a comma-delimited list of trusted domains or URLs.

For this example configuration, you would add `http://amser.provo.novell.com` to the list.

4e Click *OK*, then restart your Firefox browser.

5 In the URL field, enter the base URL of the Identity Server with port and application. For this example configuration:

```
http://amser.provo.novell.com:8080/nidp
```

The Identity Server should authenticate the user without prompting the user for authentication information. If a problem occurs, check for the following configuration errors:

- Verify the default user store and contract. See Step 13.
- View the Identity Server logging file and verify the configuration. See "Verifying the Kerberos Configuration" on page 173.
- If you make any modifications to the configuration, either in the Administration Console or to the `bcsLogin` file, restart Tomcat on the Identity Server.

6 (Conditional) If you have users who are outside the firewall, they cannot use Kerberos. SPNEGO defaults first to NTLM, then to HTTPS basic authentication. Access Manager does not support NTLM, so the NTLM prompt for username and password fails. The user is then prompted for a username and password for HTTPS basic authentication, which succeeds if the credentials are valid.

   To avoid these prompts, you can have your users enable the *Automatic logon with current user name and password* option in Internet Explorer 7.*x*. To access this option, click *Tools* > *Internet Options* > *Security* > *Custom Level*, then scroll down to *User Authentication*.

5.5 Configuring the Access Gateway for Kerberos Authentication

If you have set up a Web server that you want to require Kerberos authentication for access, you can set up a protected resource for this Web server as you would for any other Web server, and select the name of your Kerberos contract for the authentication procedure. For instructions, see "Configuring Protected Resources" in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*.

When using Kerberos for authentication, the LDAP credentials are not available. If you need LDAP credentials to provide single sign-on to some resources, see Section 4.5, "Configuring Password Retrieval," on page 150.

Defining Shared Settings

6

You can define shared settings so that they can be reused and are available in any Identity Server cluster configuration. The settings include:

- **Attribute sets:** Sets of attributes that are exchangeable between identity and service providers.
- **User matching expressions:** The logic of the query to the user store for identification when an assertion is received from an identity provider.
- **Shared Secret names:** Custom shared secret names that you want to be available when configuring policies.
- **LDAP attributes:** Custom LDAP attribute names that you want to be available when configuring policies.
- **Authentication card images:** Custom images that you can assign to authentication cards to uniquely identify an authentication procedure.

These features are configurable from the *Shared Setting* tab on the Identity Servers page.

This section describes the following tasks:

6.1 Configuring Attribute Sets

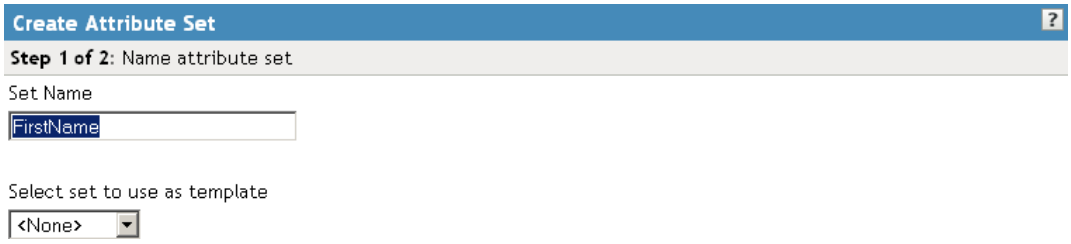
The attributes you specify on the Identity Server are used in attribute requests and responses, depending on whether you are configuring a service provider (request) or identity provider (response). Attribute sets provide a common naming scheme for the exchange. For example, an attribute set can map an LDAP attribute such as givenName to the equivalent remote name used at the service provider, which might be firstName. These shared attributes can then be used for policy enforcement, user identification, and data injection.

For example, you could have a Web server application that requires the user’s e-mail address. For this scenario, you configure the Web server to be a protected resource of the Access Gateway, and you configure an Identity Injection policy to add the user’s email address to a custom HTTP header. When the user accesses the protected resource, the value of the email attribute is retrieved. However, if you create an attribute set with this attribute, then assign it to be sent with the authentication response of the Embedded Service Provider of the Access Gateway, the value is cached at authentication and is immediately available. If you have multiple attributes that you are using in policies, obtaining the values in one LDAP request at authentication time can reduce the amount of LDAP traffic to your user store.

You can define multiple attribute sets and assign them to different trusted relationships. You can also use the same attribute set for multiple trusted relationships.

To create and configure an attribute set:

1 In the Administration Console, click *Devices* > *Identity Server* > *Shared Settings* > *Attribute Sets* > *New*.



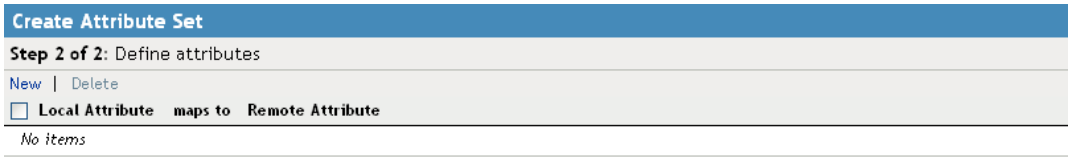
2 Fill in the following fields:

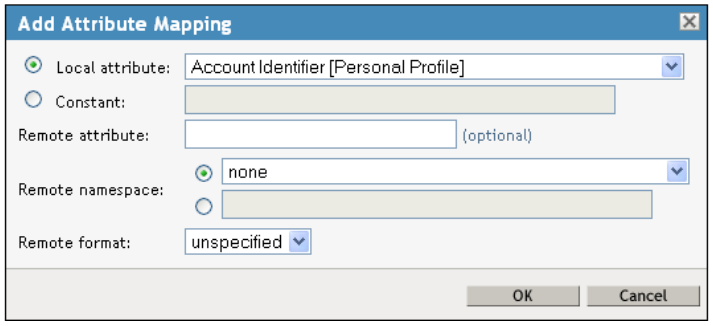
Set Name: Specify a name for identifying the attribute set.

Select set to use as template: Select an existing attribute set that you have created, which you can use as a template for the new set, or select *None*. To modify an existing attribute set, select that set as a template.

3 Click *Next*.

4 To add an attribute to the set, click *New*.





5 Fill in the following fields:

Specify the attribute. Select from the following:

- **Local Attribute:** Select an attribute from the drop-down list of all server profile, LDAP, and shared secret attributes. For example, you can select *All Roles* to use in role policies, which enables trusted providers to send role information in authentication assertions. Share secret attributes must be created before they can be added to an attribute set. For instructions, see Section 6.4.1, “Creating Shared Secret Names,” on page 182.
- **Constant:** Specify a value that is constant for all users of this attribute set. The name of the attribute that is associated with this value is specified in the *Remote Attribute* field.

Remote Attribute: Specify the name of the attribute defined at the external provider. The text for this field is case sensitive.

- A value is optional if you are mapping a local attribute. If you leave this field blank, the system sends an internal value that is recognized between Identity Servers.

  For a SAML 1.1 identity consumer (service provider), a name identifier received in an assertion is automatically given a remote attribute name of *saml:NameIdentifier*. This allows the name identifier to be mapped to a profile attribute that can then be used in policy definitions.

- A value is required if you are mapping a constant.

  An attribute set with a constant is usually set up when the Identity Server is acting as an identity provider for a SAML or Liberty service provider. The name must match the attribute name that the service provider is using.

Remote namespace: Specify the namespace defined for the attribute by the remote system:

- If you are defining an attribute set for LDAP, select *none*. If you want a service provider to accept any namespace specified by an identity provider, select *none*. If you want an identity provider to use a default namespace, select *none*. The `urn:oasis:names:tc:SAML:1.0:assertion` value is sent as the default.
- If you are defining an attribute set for CardSpace, select the following:

  `http://schema.xml/soap.org/ws/2005/05/identity/claims`
- If you are defining an attribute set for WS Federation, select the radio button next to the text box, then specify the following name in the text box.

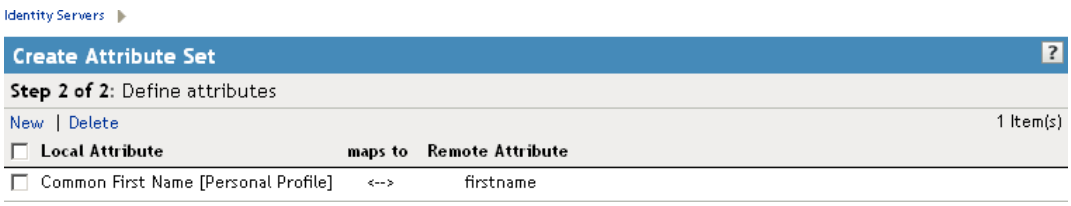
  `http://schemas.xmlsoap.org/claims`
- If you want to specify a new namespace, select the radial button by the text box, then specify the name in the text box.

Remote format: Select one of the following formats:

- **unspecified:** Indicates that the interpretation of the content is implementation-specific.
- **uri:** Indicates that the interpretation of the content is application-specific.
- **basic:** Indicates that the content conforms to the xs:Name format as defined for attribute profiles.

6 Click *OK*.

The system displays the map settings on the Define Attributes page, as shown below:



You can continue adding as many attributes as you need.

7 Click *Finish* after you created the map.

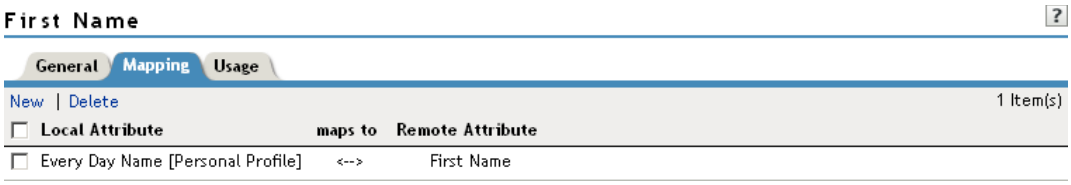
The system displays the map on the Attribute Sets page, as well as indicating whether it is in use by a provider.

8 (Conditional) To configure a provider to use the attribute set, see Section 7.6, “Selecting Attributes for a Trusted Provider,” on page 203.

6.2 Editing Attribute Sets

You can edit attribute sets that have been created in the system. (See Section 6.1, “Configuring Attribute Sets,” on page 177.)

1. In the Administration Console, click *Devices* > *Identity Server* > *Shared Settings* > *Attribute Sets*.
2. Click the name of the attribute set that you want to edit.



3. The system displays an attribute set page with the following tabs:

   General: Click to edit the name of the attribute set.

   Mapping: Click to edit the attribute map.

   Usage: Displays where the attribute set is used. Informational only.
4. Click *OK*, then click *Close*.

6.3 Configuring User Matching Expressions

When a service provider receives an assertion from a trusted identity provider, the service provider tries to identify the user. The service provider can be configured to take one of the following actions:

- Accept that the assertion contains a valid user and authenticate the user locally with a temporary identity and account. As soon as the user logs out, the account and identity are destroyed.
- Use the attributes in the assertion to match a user in the local user store. When you want the service provider to take this action, you need to create a user matching expression.
- Use the attributes in the assertion to match a user in the local user store and when the match fails, create an account (called provisioning) for the user in the local user store of the service provider. When you want the service provider to take this action, you need to create a user matching expression.

The user matching expression is used to format a query to the user store based on attributes received in the assertion from the identity provider. This query must return a match for one user.

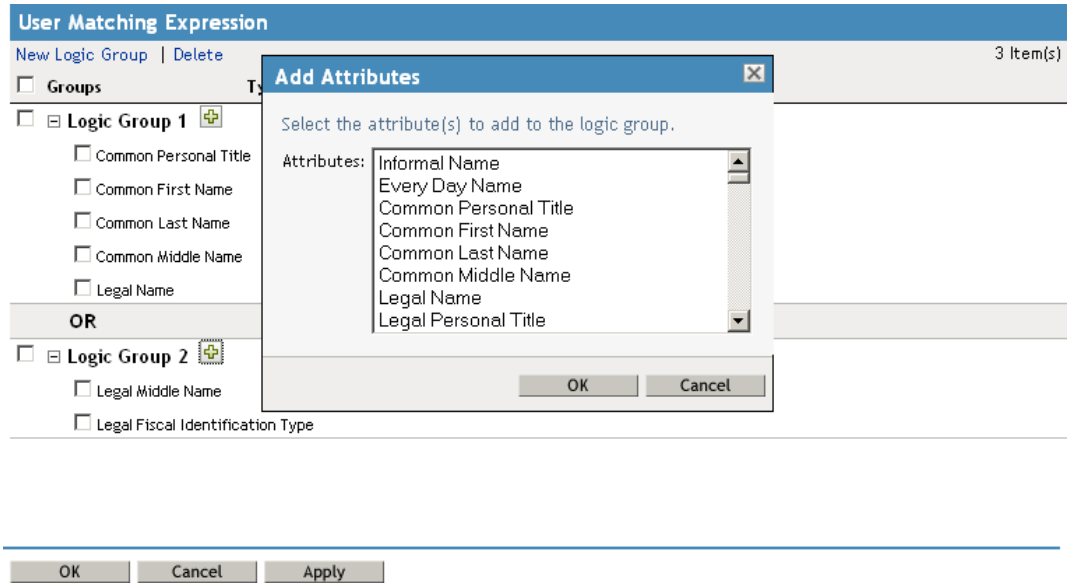
- If the query returns a match for multiple users, the request fails and the user is denied access.
- If the query fails to find a match and you have selected provisioning, the service provider uses the attributes to create an account for this user in its user store. If you haven’t selected provisioning, the request fails and the user is denied access.

The user matching expression defines the logic of the query. You must know the LDAP attributes that are used to name the users in the user store in order to create the user’s distinguished name and uniquely identify the users.

For example, if the service provider user store uses the email attribute to identify users, the identity provider should be configured to send the email attribute. The service provider would use this attribute in a user matching expression to find the user in the user store. If a match is found, the user is granted access. If the user is not found, that attribute can be used to create an account for the user. The assertion must contain all the attributes that the user store requires to create an account.

To create a user matching expression:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Shared Settings* > *User Matching Expressions*.
2. Click *New*, or click the name of an existing user matching expression.
3. Specify a name for the user lookup expression.
4. Click the *Add Attributes* icon (plus sign), then select attributes to add to the logic group. (Use the Shift key to select several attributes.)



5. Click *OK*.
6. To add logic groups, click *New Logic Group*.

   The *Type* drop-down (AND or OR) applies only between groups. Attributes within a group are always the opposite of the type selection. For example, if the *Type* value is AND, the attributes within the group are OR.
7. Click the *Add Attributes* icon (plus sign) to add attributes to the next logic group, then click *OK*.
8. Click *Finish*.
9. (Conditional) If you selected attributes from the Custom, Employee, or Personal profile, you need to enable the profile so that the attribute can be shared:
   - **9a** Click *Servers* > *Edit* > *Liberty* > *Web Service Provider*.
   - **9b** Select the profiles that need to be enabled, then click *Enable*.
   - **9c** Click *OK*, then update the Identity Server.

6.4 Adding Custom Attributes

You can add custom shared secret names or LDAP attribute names that you want to make available for selection when setting up policies.

6.4.1 Creating Shared Secret Names

The shared secret consists of a secret name and one or more secret entry names. You can create a secret name only, or a secret name and an entry name. For ease of use, the entry name should match the policy that uses it:

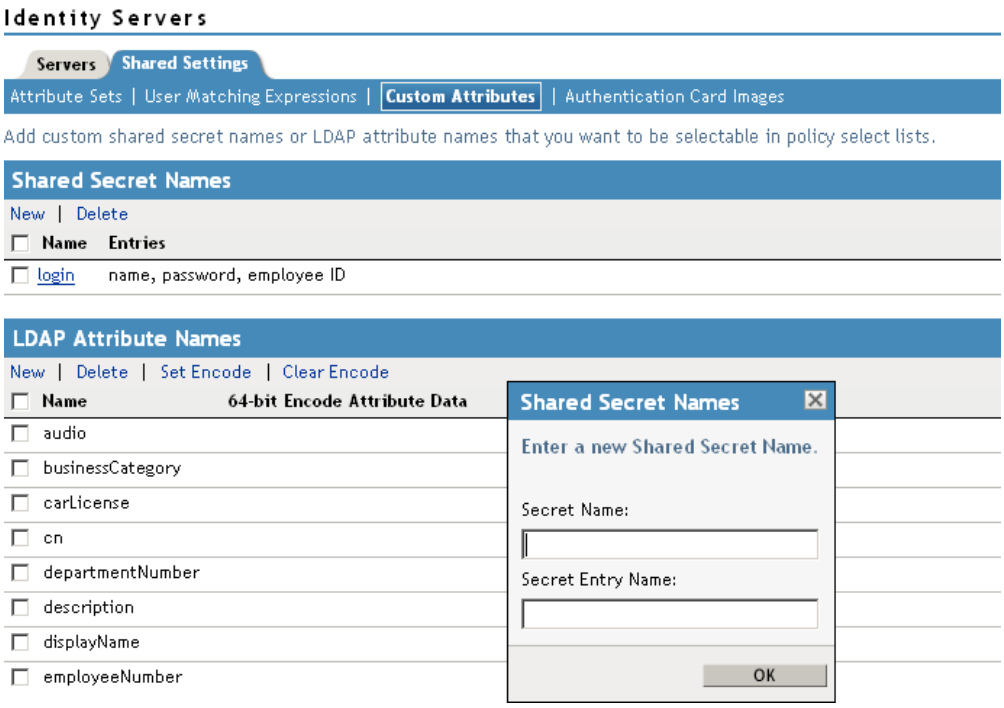
- For a Form Fill policy, the entry name should match a form field name.
- For an Identity Injection policy, the entry name should match the Custom Header Name.

For more information on how to use shared secrets with policies, see “Creating and Managing Shared Secrets” in the *Novell Access Manager 3.1 SP2 Policy Guide*.

The Identity Server needs to be configured to use shared secrets. For information on this process, see Section 3.1.4, “Configuring a User Store for Secrets,” on page 109.

Shared secret names can be created either on the Custom Attributes page or in the associated policy that consumes them.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Shared Settings* > *Custom Attributes*.
2. To create shared secret names, click *New*.



3 Enter a new shared secret name and, optionally, a secret entry name.

4 Click *OK*.

5 (Optional) To create additional entries for the secret, click the name of the secret, click *New*, specify an entry name, then click *OK*.

WARNING: The Identity Server currently has no mechanism to determine whether a secret is being used by a policy. Before you delete a shared secret, you must make sure it is not being used.

6.4.2 Creating LDAP Attribute Names

LDAP attributes are available for all policies. LDAP attribute names can be created either on the Custom Attributes page or in the associated policy that consumes them. The attribute names that you specify must match the name of an attribute of the user class in your LDAP user store.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Shared Settings* > *Custom Attributes*.

This list contains the attributes for the inetOrgPerson class. It is customizable.

audio: Uses a u-law encoded sound file that is stored in the directory.

businessCategory: Describes the kind of business performed by an organization.

carLicense: Vehicle license or registration plate.

cn: The X.500 commonName attribute, which contains a name of an object. If the object corresponds to a person, it is typically the person's full name.

departmentNumber: Identifies a department within an organization.

displayName: The preferred name of a person to be used when displaying entries. When displaying an entry, especially within a one-line summary list, it is useful to use this value. Because other attribute types such as cn are multivalued, an additional attribute type is needed.

employeeNumber: Numerically identifies a person within an organization.

employeeType: Identifies the type of employee.

givenName: Identifies the person's name that is not his or her surname or middle name.

homePhone: Identifies a person by home phone.

homePostalAddress: Identifies a person by home address.

initials: Identifies a person by his or her initials. This attribute contains the initials of an individual, but not the surname.

jpegPhoto: Stores one or more images of a person, in JPEG format.

labeledURI: Uniform Resource Identifier with an optional label. The label describes the resource to which the URI points.

mail: A user's e-mail address.

manager: Identifies a person as a manager.

mobile: Specifies a mobile telephone number associated with a person.

o: The name of an organization.

pager: The pager telephone number for an object.

photo: Specifies a photograph for an object.

preferredLanguage: Indicates an individual's preferred written or spoken language.

roomNumber: The room number of an object.

secretary: Specifies the secretary of a person.

sn: The X.500 surname attribute, which contains the family name of a person.

uid: User ID.

userCertificate: An attribute stored and requested in the binary form.

userPKCS12: A format to exchange personal identity information. Use this attribute when information is stored in a directory service.

userSMIMECertificate: PKCS#7 SignedData used to support S/MIME. This value indicates that the content that is signed is ignored by consumers of userSMIMECertificate values.

x500uniqueIdentifier: Distinguishes between objects when a distinguished name has been reused. This is a different attribute type from both the *uid* and the *uniqueIdentifier* type.

2 To add a name:

   2a Click *New*.

   2b If you want the attribute to return raw data rather binary data, select *64-bit Encode Attribute Data*.

   2c Click *OK*.

3 To modify the 64-bit attribute data encoding, click an attribute's check box, then click one of the following links:

   Set Encode: Specifies that LDAP returns a raw format of the attribute rather than binary format, which Access Manager encodes to base64, so that the protected resource understands the attribute. You might use base64 encoding if you use certificates that require raw bites rather than binary string format.

   Clear Encode: Deletes the 64-bit data encoding setting.

4 Click *Apply* to save changes, then click the *Servers* tab to return to the Servers page.

6.5 Adding Authentication Card Images

Each authentication contract, CardSpace card, and managed card template must have a card associated with it.

To add new images, the image files must be available from the workstation where you are authenticated to the Administration Console. Images must fall within the size bounds of 60 pixels wide by 45 pixels high through 200 pixels wide by 150 pixels high.

To add a card image:

1 In the Administration Console, click *Devices* > *Identity Servers* > *Shared Settings* > *Authentication Card Images*.

2 Click *New*.

3 Fill in the following fields.

   Name: Specify a name for the image.

   Description: Describe the image and its purpose.

   File: Click *Browse*, locate the image file, then click *Open*.

Locale: From the drop-down menu, select the language for the card or select *All Locales* if the card can be used with all languages.

4 Click *OK*.

5 If you did not specify *All Locales* for the *Locale,* continue with Section 6.6, "Creating an Image Set," on page 185.

6.6 Creating an Image Set

You can create card images for specific locales as well as a default image for all locales. The images need to be placed in an image set that allows the browser to display the image associated with the requested locale. If the browser requests a locale for which you have not defined an image, the *All Locales* image is displayed. If an *All Locales* image is not available, the browser displays the *Image not Available* card.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Shared Settings* > *Authentication Card Images*.

2 Click the card image.

3 To add an image to the set, click *New*, then fill in the following fields:

File: Click *Browse*, then browse to the location of the file.

Locale: Select the locale from the drop-down menu.

4 Click *OK*.

Configuring SAML and Liberty Trusted Providers

7

This section discusses configuring trust so that two user accounts can be associated with each other without the sites exchanging user data. It explains how to use the Liberty, SAML 1.1, and SAML 2.0 protocols to set up the trust with internal and external identity providers, service providers, and Embedded Service Providers (ESPs).

About SAML and Liberty

For information about how Access Manager uses SAML, see Appendix B, “Understanding How Access Manager Uses SAML,” on page 369.

For conceptual information about Liberty, see Appendix A, “About Liberty,” on page 367.

7.1 Understanding the Trust Model

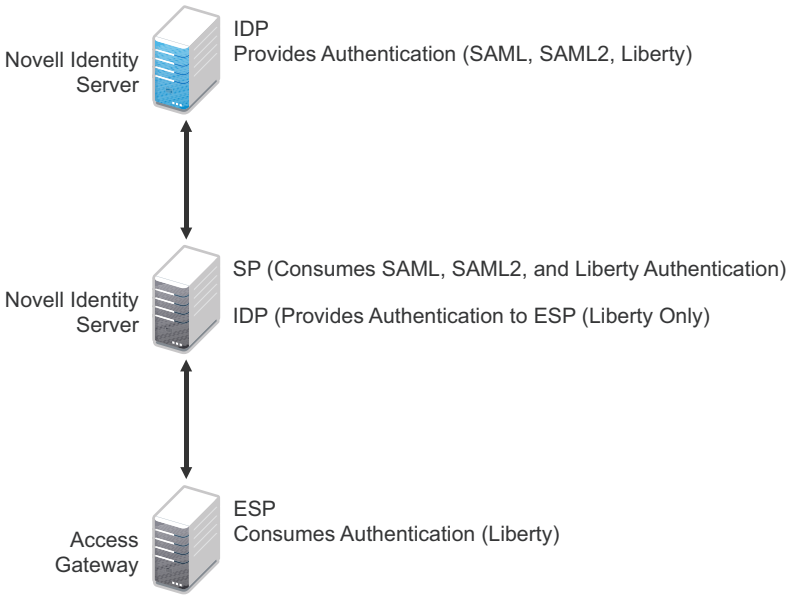
Setting up trust involves system administrators agreeing on how to establish a secure method for providing and consuming authentication assertions between their Identity Servers. An Identity Server is always installed as an identity provider, which is used to provide authentication to trusted service providers and Embedded Service Providers (ESPs). It can also be configured to be a service provider and trust the authentication of an identity provider.

7.1.1 Identity Providers and Consumers

An Identity Server can be configured as an identity provider, which allows other service providers to trust it for authentication. It can also be configured as a service provider, which enables the Identity Server to consume authentication assertions from trusted identity providers. Figure 7-1 depicts two

Identity Servers. The Identity Server at the top of the figure is configured as an identity provider for SAML 1.1, SAML 2.0, and Liberty authentication. The Identity Server in the middle of the figure is configured as a service provider, consuming the authentication credentials of the top Identity Server. This second Identity Server is also configured as an identity provider, providing authentication for the Embedded Service Provider of the Access Gateway.

Figure 7-1 *Identity Server Trust*

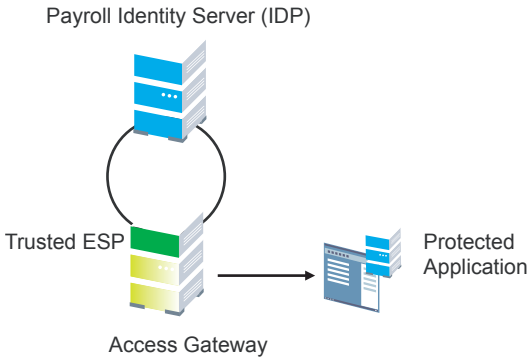


As an administrator, you determine whether your server is to be used as the identity provider or service provider in the trust relationship. You and the trusted partner agree to exchange identity provider metadata, and then you create references to the trusted partner's identity provider or service provider in your Identity Server configuration. You can obtain metadata via a URL or an XML document, then enter it in the system when you create the reference.

7.1.2 Embedded Service Providers

In addition to setting up trust with internal or external service providers, you can reference Embedded Service Providers (ESPs) in your enterprise. An ESP uses the Liberty protocol and does not require metadata entry, because this exchange happens automatically. The ESP comes with Access Manager and is embedded in the Access Gateways, the J2EE agents, and a version of the SSL VPN server. The ESP facilitates authentication between the Identity Server and the resource protected by the device, as shown in as shown in Figure 7-2.

Figure 7-2 *Embedded Service Provider*



7.1.3 Configuration Overview

The following high-level tasks describe the process required to set up the trust model between an identity provider and a service provider. Although these tasks assume that both providers are Identity Servers provided with Access Manager, similar tasks must be performed when one of the providers is a third-party application.

1. Administrators at each company install and configure the Identity Server.

   See Section 1.1.1, “Creating a Cluster Configuration,” on page 16. (You should already be familiar with the *Novell Access Manager 3.1 SP2 Installation Guide*.)

2. Administrators at each company must import the trusted root certificate of the other Identity Server into the NIDP trust store.

   Click *Devices* > *Identity Servers* > *Servers* > *Edit* > *Security* > *NIDP Trust Store*, then auto import the certificate. Use the SSL port (8443) even if you haven’t set up the base URL of the Identity Server to use HTTPS.

3. Administrators must exchange Identity Server metadata with the trusted partner.

   Metadata is generated by the Identity Server and can be obtained via a URL or an XML document, then entered in the system when you create the reference. This step is not applicable if you are referencing an ESP. When you reference an ESP, the system lists the installed ESPs for you to choose, and no metadata entry is required.

4. Create the reference to the trusted identity provider and the service provider.

   This procedure associates the metadata with the new provider. See Section 7.3.1, “Creating a Trusted Provider for Liberty or SAML 2.0,” on page 194.

5. Configure user authentication.

   This procedure defines how your Identity Server interacts with the trusted provider during user authentication. Access Manager comes with default basic authentication settings already enabled. See Chapter 11, “Configuring User Identification Methods for Federation,” on page 281.

   Additional important steps for enabling authentication between trusted providers include:

   - Setting up the necessary authentication contracts. See Section 3.4, “Configuring Authentication Contracts,” on page 125.

- Enabling the profiles that you are using. See Section 13.2, “Managing Web Services and Profiles,” on page 296.
- Enabling the *Always Allow Interaction* option on the Web Service Consumer page. See Section 13.5, “Configuring the Web Service Consumer,” on page 307.

6. (Conditional) If you are setting up SAML 1.1 federation, the protocol does not allow the target link after federation to be automatically configured. You must manually configure this setting.

   See “Specifying the Intersite Transfer Service URL for the Login URL Option” on page 223.

NOTE: For a tutorial that explains all the steps for setting up federation between two Novell Identity Servers, see “Setting Up Federation” in the *Novell Access Manager 3.1 SP2 Setup Guide*.

7.2 Configuring General Provider Options

The following options are global because they affect any identity providers or identity consumers (service providers) that the Identity Server has been configured to trust:

7.2.1 Configuring the General Identity Provider Options

The following options affect all identity providers that the Identity Server has been configured to trust.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Identity Providers*.

2 To specify identity provider settings, fill in the following fields:

Show logged out providers: Displays logged-out providers on the identity provider’s logout confirmation page.

Require Signed Authentication Requests: Specifies that for the Liberty 1.2 and SAML 2.0 protocols, authentication requests from service providers must be signed. When you enable this option for the identity provider, you must also enable the *Sign Authentication Requests* option under the *Identity Consumer* heading on this page for the external trusted service provider.

Use Introductions (Publish Authentications): Enables single sign-on from the service provider to the identity provider. The service provider determines the identity providers that users are already logged into, and then selectively and automatically asks for authentication from one of the identity providers. Introductions are enabled only between service and identity providers that have agreed to a circle of trust, which means that they have agreed upon a common domain name for this purpose.

After authenticating a user, the identity provider accesses a service at the service domain and writes a cookie to the common part of the service domain, publishing that the authentication has occurred.

- **Service Domain (Local and Common):** Enables a service provider to access a service at the service domain prior to authenticating a user. This service reads cookies obtained at this domain and discovers if any identity providers have provided authentication to the

user. The service provider determines whether any of these identity providers can authenticate a user without credentials. The service domain must resolve to the same IP address as the base URL domain.

For example, if an agreed-upon common domain is *xyz.com*, the service provider can specify a service domain of *sp.xyz.com*, and the identity provider can specify a service domain of *idp.xyz.com*. For the identity provider, *xyz.com* is the common value entered, and *idp* is the local value.

- **Port:** The port to use for identity provider introductions. Port 8445 for HTTPS is the default and must be opened on your firewall. If you specify a different port, you must edit the Tomcat `server.xml` file.

SSL Certificate: Displays the Keystore page that you use to locate and replace the test-provider SSL certificate for this configuration.

The Identity Server comes with a test-provider certificate that you must replace for your production environment. This certificate is used for identity provider introductions. You can replace the test certificate now or after you have configured the Identity Server. If you create the certificate and replace the test-connector now, you can save some time by restarting Tomcat only once. Tomcat must be restarted whenever you assign an Identity Server to a configuration and whenever you update a certificate key store. See Section 1.3.3, "Managing the Keys, Certificates, and Trust Stores," on page 29.

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

7.2.2 Configuring the General Identity Consumer Options

The following options affect all identity consumers (service providers) that the Identity Server has been configured to trust.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Identity Consumer*.

2 Specify whether the Identity Server can run as an identity consumer.

When the Identity Server is configured to run as an identity consumer, the Identity Server can receive (consume) authentication assertions from other identity providers.

Enable: Enables this site to function as service provider. This setting is enabled by default.

If this option is disabled, the Identity Server cannot trust or consume authentication assertions from other identity providers. You can create and enable identity providers for the various protocols, but they are not loaded or used until this option is enabled.

Require Signed Assertions: Specifies that all SAML assertions received by the service provider are signed by the issuing SAML authority. The signing authority uses a key pair to sign SAML data sent to this trusted provider.

Sign Authentication Requests: Specifies that the service provider signs authentication requests sent to an identity provider when using the Liberty 1.2 and SAML 2.0 protocols.

Use Introductions (Discover IDP Authentications): Enables a service provider to discover whether a user has authenticated to a trusted identity provider, so the user can use single sign-on without requiring authentication credentials.

- **Service domain:** The shared, common domain for all providers in the circle of trust. This domain must resolve to the same IP address as the base URL domain. You must enable the *Identity Consumer* option to enable this field.

- **Port:** The port to use for identity consumer introductions. Port 8446 for HTTPS is the default and must be opened on your firewall. If you specify a different port, you must edit the Tomcat `server.xml` file.

IMPORTANT: If you enable the *Use Introductions* option and you want to allow your users to select which identity provider to use for authentication rather than use single sign-on, you need to configure the Introductions class. See Section 7.2.3, "Configuring the Introductions Class," on page 192.

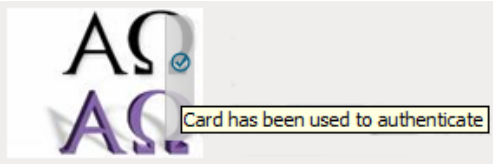
SSL Certificate: Displays the Keystore page that you use to locate and replace the test-consumer SSL certificate for this configuration.

The Identity Server comes with a test-consumer certificate that you must replace for your production environment. This certificate is used for identity consumer introductions. You can replace the test certificate now or after you have configured the Identity Server. If you create the certificate and replace the test-connector now, you can save some time by restarting Tomcat only once. Tomcat must be restarted whenever you assign an Identity Server to a configuration and whenever you update a certificate key store. See Section 1.3.3, "Managing the Keys, Certificates, and Trust Stores," on page 29.

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

7.2.3 Configuring the Introductions Class

The Introduction class determines whether the user can select an identity provider to trust when the Identity Server is acting as a service provider. The default behavior is for introductions to happen automatically, thus allowing single sign-on. The Identity Server passively checks with the identity providers, one at time, to see if they can authenticate the service provider. If the identity provider can authenticate the user and the Introductions class is enabled, the user is presented with one or more cards that look similar to the following:



The small check mark indicates to the user that this is a possible card. When the user mouses over the card, the description appears. If the user selects one of these cards, the user is automatically authenticated.

To configure the Introductions class:

1 In the Administration Console, click *Devices* > *Identity Server* > *Servers* > *Edit* > *Local* > *Classes* > *Introductions*.

2 Click *Properties* > *New*, then specify the following values.

Property Name: Specify `ShowUser`.

Property Value: Specify `true`.

3 Click *OK*.

4 Return to the Servers page, then update the *Identity Server*.

5 When you configure this class, you need to also enable the *Use Introductions* option. Continue with Section 7.2.2, “Configuring the General Identity Consumer Options,” on page 191.

7.2.4 Configuring the Trust Levels Class

The Trust Levels class allows you to specify an authentication level or rank for class types that do not appear on the Defaults page and for which you have not defined a contract. The level is used to rank the requested type. Using the authentication level and the comparison context, the Identity Server can determine whether any contracts meet the requirements of the request. If one or more contracts match the request, the user is presented with the appropriate authentication prompts. For more information and other configuration options, see Section 3.5, “Specifying Authentication Defaults,” on page 131 and Section 3.5.1, “Specifying Authentication Types,” on page 132

1 In the Administration Console, click *Devices* > *Identity Server* > *Servers* > *Edit* > *Local* > *Classes* > *Trust Levels*.

2 Click *Properties* > *New*, then specify the following values.

Property Name: Specify `SetClassTrustLevels`.

Property Value: Specify `true`.

3 For each class type for which you want to set a level, create a property for that class.

3a Set the *Property Name* to the name of the class. For example, use one of the following:

```
urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol
```

For additional values, refer to the SAML2 and Liberty Authentication Context Specifications.

3b Set the *Property Value* to the security level or rank you want for the class. A level of 2 is higher than a level of 1.

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

7.3 Managing Trusted Providers

The procedure for establishing trust between providers begins with obtaining metadata for the trusted provider. If you are using the Novell Identity Server, protocol-specific metadata is available via a URL.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Servers* > *Edit* > *[Protocol]*.

For the protocol, select Liberty, SAML 1.1 or SAML 2.0.

2 Select one of the following actions:

New: Launches the Create Trusted Identity Provider Wizard or the Create Trusted Service Provider Wizard, depending on your selection. See one of the following for more information:.

- Section 7.3.1, “Creating a Trusted Provider for Liberty or SAML 2.0,” on page 194
- Section 7.3.2, “Creating a Trusted Service Provider for SAML 1.1,” on page 196
- Section 7.3.3, “Creating a Trusted Identity Provider for SAML 1.1,” on page 198

Delete: Allows you to delete the selected identity or service provider.

Enable: Enables the selected identity or service provider.

Disable: Disables the selected identity or service provider. When a provider is disabled, the server does not load the definition. The definition is not deleted, and at a future time, the provider can be enabled.

IMPORTANT: When selecting which protocol to use, be aware of logout behavior of the SAML 1.1 protocol. The SAML 2.0 and Liberty 1.2 protocols define a logout mechanism whereby the service provider sends a logout command to the trusted identity provider when a user logs out at a service provider. SAML 1.1 does not provide such a mechanism. For this reason, when a logout occurs at the SAML 1.1 service provider, no logout occurs at the trusted identity provider. A valid session is still running at the identity provider, and no credentials need to be entered. In order to log out at both providers, the user must navigate to the identity provider that authenticated him to the SAML 1.1 service provider and log out manually.

7.3.1 Creating a Trusted Provider for Liberty or SAML 2.0

You can configure the Identity Server to trust a service provider or an identity provider.

- When you create a trusted identity provider, you are allowing that identity provider to authenticate the user and the Identity Server acts as a service provider.
- When you create a trusted service provider, you are configuring the Identity Server to provide authentication for the service provider and the Identity Server acts as an identity provider.

Both of these types of trust relationships require the identity provider to establish a trusted relationship with the service provider and the service provider to establish a trusted relationship with the identity provider.

Prerequisites

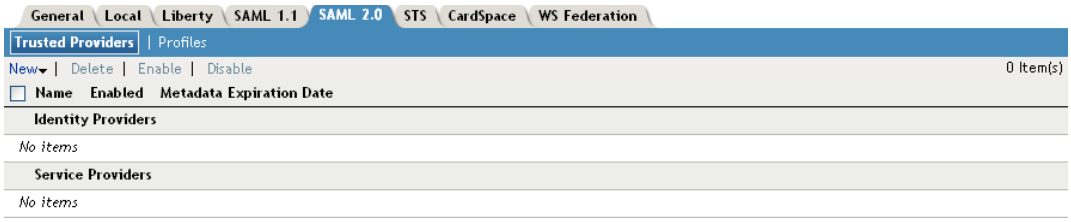
Before you can create a trusted provider, you must complete the following tasks:

- Imported the trusted root of the provider's SSL certificate into the NIDP trust store. For instructions, see Section 1.3.3, "Managing the Keys, Certificates, and Trust Stores," on page 29.
- Shared the trusted root of the SSL certificate of your Identity Server with the other provider so that the administrator can imported it into the provider's trust store.
- Obtained the metadata URL from the other provider or an XML file with the metadata.
- Shared the metadata URL of your Identity Server with the other provider or sent an XML file with the metadata.
- Enabled the protocol. Click *Devices* > *Identity Servers* > *Edit*, and on the Configuration page, verify that the required protocol in the Enabled Protocols section has been enabled.

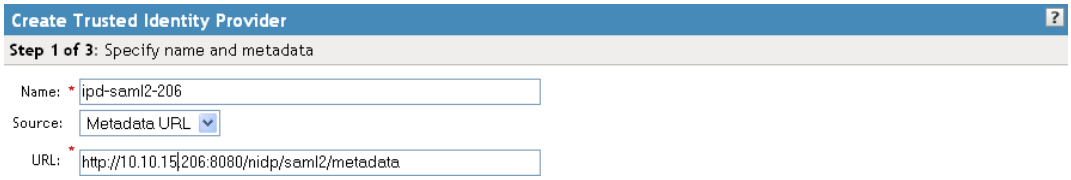
Procedure

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *[Protocol]*.

For the protocol, click *Liberty* or *SAML 2.0*.



2 Click *New*, then click *Identity Provider* or *Service Provider*.



3 In the *Name* option, specify a name by which you want to refer to the provider.

4 Select one of the following sources for the metadata:

Metadata URL: Specify the metadata URL for a trusted provider. The system retrieves protocol metadata by using the specified URL.

Examples of metadata URLs for an Identity Server acting as a trusted provider with an IP address 10.1.1.1:

- **Liberty:** `http://10.1.1.1:8080/nidp/idff/metadata`
- **Liberty:** `https://10.1.1.1:8443/nidp/idff/metadata`
- **SAML 2.0:** `http://10.1.1.1:8080/nidp/saml2/metadata`
- **SAML 2.0:** `https://10.1.1.1:8443/nidp/saml2/metadata`

The default values nidp and 8080 are established during product installation; nidp is the Tomcat application name. If you have set up SSL, you can use https and port 8443.

If your Identity Server and Administration Console are on different machines, use HTTP to import the metadata. If you are required to use HTTPS with this configuration, you must import the trusted root certificate of the provider into the trust store of the Administration Console. You need to use the Java `keytool` to import the certificate into the `cacerts` file in the security directory of the Administration Console.

Linux: `/opt/novell/java/jre/lib/security`

Windows Server 2003: `\Program Files\Novell\jre\lib\security`

Windows Server 2008: `\Program Files (x86)\Novell\jre\lib\security`

If you do not want to use HTTP and you do not want to import a certificate into the Administration Console, you can use the *Metadata Text* option. In a browser, enter the HTTP URL of the metadata. View the text from the source page, save the source metadata, then paste it into the *Metadata Text* option.

Metadata Text: An editable field in which you can paste copied metadata text from an XML document, assuming you obtained the metadata via e-mail or disk and are not using a URL. If you copy metadata text from a Web browser, you must copy the text from the page source.

5 Click *Next*.

6 Review the metadata certificates, then select one of the following actions:

- For a service provider, continue with Step 8.
- For an identity provider, click *Next*, then continue with Step 7.

7 (Identity Provider only) Configure an authentication card to use with this identity provider. Fill in the following fields:

ID: (Optional) Specify an alphanumeric value that identifies the card. If you need to reference this card outside of the Administration Console, you need to specify a value here. If you do not assign a value, the Identity Server creates one for its internal use

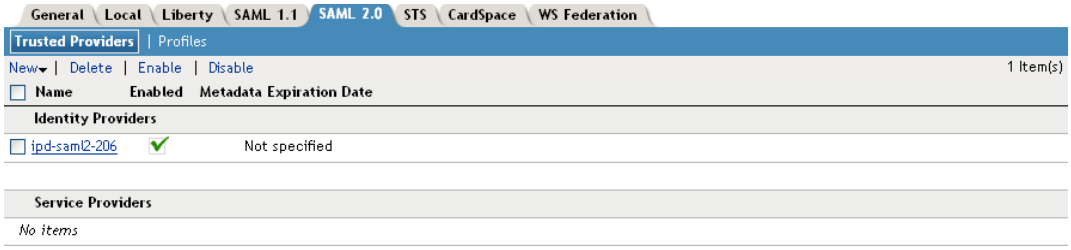
Text: Specify the text that is displayed on the card to the user.

Image: Specify the image to be displayed on the card. Select the image from the drop down list. To add an image to the list, click <*Select local image*>.

Show Card: Determine whether the card is shown to the user, which allows the user to select and use the card for authentication. If this option is not selected, the card is only used when a service provider makes a request for the card.

Passive Authentication Only: Select this option if you do not want the Identity Server to prompt the user for credentials. If the user has already authenticated and the credentials satisfy the requirements of this contract, the user is passively authenticated. If the user's credentials do not satisfy the requirements of this contract, the user is denied access.

8 Click *Finish*. The system displays the trusted provider on the protocol page.



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

The wizard allows you to configure the required options and relies upon the default settings for the other federation options. For information about how to configure the default settings and how to configure the other available options, see Section 7.4, "Modifying a Trusted Provider," on page 199.

7.3.2 Creating a Trusted Service Provider for SAML 1.1

Before you can create a trusted service provider, you must complete the following tasks:

- Imported the trusted root of the provider's SSL certificate into the NIDP trust store. For instructions, see Section 1.3.3, "Managing the Keys, Certificates, and Trust Stores," on page 29.
- Shared the trusted root of the SSL certificate of your Identity Server with the service provider so that the administrator can imported it into the service provider's trust store.
- Obtained the metadata URL from the service provider, an XML file with the metadata, or the information required for manual entry. For more information about the manual entry option, see Section 7.7.4, "Editing a SAML 1.1 Service Provider's Metadata," on page 209.

- Shared the metadata URL of your Identity Server with the service provider or an XML file with the metadata.
- Enabled the protocol. Click *Devices* > *Identity Servers* > *Edit*, and on the Configuration page, verify that the required protocol in the Enabled Protocols section has been enabled.

To create a service provider:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 1.1*.
2. Click *New*, then click *Service Provider*.
3. In the *Name* option, specify a name by which you want to refer to the provider.
4. Select one of the following sources for the metadata:

   Metadata URL: Specify the metadata URL for a trusted provider. The system retrieves protocol metadata using the specified URL. Examples of metadata URLs for an Identity Server acting as a service provider with an IP address of 10.1.1.1:

   ```
   http://10.1.1.1:8080/nidp/saml/metadata
   https://10.1.1.1:8443/nidp/saml/metadata
   ```

   The default values nidp and 8080 are established during product installation; nidp is the Tomcat application name. If you have set up SSL, you can use https and port 8443.

   If your Identity Server and Administration Console are on different machines, use HTTP to import the metadata. If you are required to use HTTPS with this configuration, you must import the trusted root certificate of the provider into the trust store of the Administration Console. You need to use the Java `keytool` to import the certificate into the `cacerts` file in the security directory of the Administration Console.

   Linux: `/opt/novell/java/jre/lib/security`

   Windows Server 2003: `\Program Files\Novell\jre\lib\security`

   Windows Server 2008: `\Program Files (x86)\Novell\jre\lib\security`

   If you do not want to use HTTP and you do not want to import a certificate into the Administration Console, you can use the *Metadata Text* option. In a browser, enter the HTTP URL of the metadata. View the text from the source page, save the source metadata, then paste it into the *Metadata Text* option.

   Metadata Text: An editable field in which you can paste copied metadata text from an XML document, assuming you obtained the metadata via e-mail or disk and are not using a URL. If you copy metadata text from a Web browser, you must copy the text from the page source.

   Manual Entry: Allows you to enter metadata values manually. When you select this option, the system displays the Enter Metadata Values page. See "Editing a SAML 1.1 Service Provider's Metadata" on page 209.
5. Click *Next*.
6. Review the metadata certificates, then click *Finish*.
7. Click *OK*, then update the Identity Server.

   The wizard allows you to configure the required options and relies upon the default settings for the other options. For information about how to configure the default settings and how to configure the other available options, see Section 7.4, "Modifying a Trusted Provider," on page 199.

7.3.3 Creating a Trusted Identity Provider for SAML 1.1

Before you can create a trusted identity provider, you must complete the following tasks:

- Imported the trusted root of the provider's SSL certificate into the NIDP trust store. For instructions, see Section 1.3.3, "Managing the Keys, Certificates, and Trust Stores," on page 29.
- Shared the trusted root of the SSL certificate of your Identity Server with the identity provider so that the administrator can imported it into the identity provider's trust store.
- Obtained the metadata URL from the identity provider, an XML file with the metadata, or the information required for manual entry. For more information about the manual entry option, see Section 7.7.3, "Editing a SAML 1.1 Identity Provider's Metadata," on page 208.
- Shared the metadata URL of your Identity Server with the identity provider or an XML file with the metadata.
- Enabled the protocol. Click *Devices* > *Identity Servers* > *Edit*, and on the Configuration page, verify that the required protocol in the Enabled Protocols section has been enabled.

To create an identity provider:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Servers* > *Edit* > *SAML 1.1*.
2. Click *New*, then click *Identity Provider*.
3. In the *Name* option, specify a name by which you want to refer to the provider.
4. Select one of the following sources for the metadata:

   Metadata URL: Specify the metadata URL for a trusted provider. The system retrieves protocol metadata using the specified URL. Examples of metadata URLs for an Identity Server acting as an identity provider with an IP address of 10.1.1.1:

   ```
   http://10.1.1.1:8080/nidp/saml/metadata
   https://10.1.1.1:8443/nidp/saml/metadata
   ```

   The default values nidp and 8080 are established during product installation; nidp is the Tomcat application name. If you have set up SSL, you can use https and port 8443.

   If your Identity Server and Administration Console are on different machines, use HTTP to import the metadata. If you are required to use HTTPS with this configuration, you must import the trusted root certificate of the provider into the trust store of the Administration Console. You need to use the Java `keytool` to import the certificate into the `cacerts` file in the security directory of the Administration Console.

   Linux: `/opt/novell/java/jre/lib/security`

   Windows Server 2003: `\Program Files\Novell\jre\lib\security`

   Windows Server 2008: `\Program Files (x86)\Novell\jre\lib\security`

   If you do not want to use HTTP and you do not want to import a certificate into the Administration Console, you can use the *Metadata Text* option. In a browser, enter the HTTP URL of the metadata. View the text from the source page, save the source metadata, then paste it into the *Metadata Text* option.

   Metadata Text: An editable field in which you can paste copied metadata text from an XML document, assuming you obtained the metadata via e-mail or disk and are not using a URL. If you copy metadata text from a Web browser, you must copy the text from the page source.

Manual Entry: Allows you to enter metadata values manually. When you select this option, the system displays the Enter Metadata Values page. See "Editing a SAML 1.1 Identity Provider's Metadata" on page 208.

5 Click *Next*.

6 Review the metadata certificates, then click *OK*.

7 Configure an authentication card to use with this identity provider. Fill in the following fields:

ID: (Optional) Specify an alphanumeric value that identifies the card. If you need to reference this card outside of the Administration Console, you need to specify a value here. If you do not assign a value, the Identity Server creates one for its internal use

Text: Specify the text that is displayed on the card to the user.

Login URL: Specify an Intersite Transfer Service URL.The URL has the following format, where idp.sitea.novell.com is the DNS name of the identity provider and idp.siteb.novell.com is the name of the service provider:

```
https://idp.sitea.novell.com:8443/nidp/saml/idpsend?PID=https://
idp.siteb.novell.com:8443/nidp/saml/metadata&TARGET=https://
idp.siteb.novell.com:8443/nidp/app
```

For more information, see "Specifying the Intersite Transfer Service URL for the Login URL Option" on page 223.

Image: Specify the image to be displayed on the card. Select the image from the drop down list. To add an image to the list, click <*Select local image*>.

Show Card: Determine whether the card is shown to the user, which allows the user to select and use the card for authentication. If this option is not selected, the card is only used when a service provider makes a request for the card.

8 Click *Finish*. The system displays the trusted provider on the protocol page.

9 Update the Identity Server.

The wizard allows you to configure the required options and relies upon the default settings for the other options. For information about how to configure the default settings and how to configure the other available options, see Section 7.4, "Modifying a Trusted Provider," on page 199.

7.4 Modifying a Trusted Provider

The following sections describe the configuration options available for identity providers and service providers:

You can modify the following features of an identity provider:

- **Communication Security:** See Section 7.5, "Configuring Communication Security," on page 200.
- **Attributes to Obtain at Authentication:** See Section 7.6.1, "Configuring the Attributes Obtained at Authentication," on page 204.
- **Metadata:** See Section 7.7, "Managing Metadata," on page 207.
- **Authentication Request:** See Section 7.8, "Configuring an Authentication Request for an Identity Provider," on page 211.
- **User Identification:** See Chapter 11, "Configuring User Identification Methods for Federation," on page 281.

- **Authentication Card:** See Section 7.10, “Managing the Authentication Card of an Identity Provider,” on page 220.

You can modify the following features of a service provider:

- **Communication Security:** See Section 7.5, “Configuring Communication Security,” on page 200.
- **Attributes to Send in the Response:** See Section 7.6.2, “Configuring the Attributes Sent with Authentication,” on page 205.
- **Intersite Transfer Service:** See “Configuring an Intersite Transfer Service Target for a Service Provider” on page 225.
- **Metadata:** See Section 7.7, “Managing Metadata,” on page 207.
- **Authentication Response:** See Section 7.9, “Configuring an Authentication Response for a Service Provider,” on page 216.

7.5 Configuring Communication Security

The communication security settings control the direct communication between the Identity Server and a trusted provider across the SOAP back channel. You can secure this channel with one of three methods:

Message Signing: This is the default method, and the Identity Server comes with a test signing certificate that is used to sign the back-channel messages. We recommend replacing this test signing certificate with a certificate from a well-known certificate authority. This method is secure, but it is CPU intensive. For information on replacing the default certificate, see Section 1.3.3, “Managing the Keys, Certificates, and Trust Stores,” on page 29.

Mutual SSL: This method is probably the fastest method, and if you are fine-tuning your system for performance, you should select this method. However, it requires the exchange of trusted root certificates between the Identity Server and the trusted provider. This exchange of certificates is a requirement for setting up the trust relationship between the two providers. To verify that you have exchanged certificates, see Section 1.3.3, “Managing the Keys, Certificates, and Trust Stores,” on page 29.

Basic Authentication: This method is as fast as mutual SSL and the least expensive because it doesn’t require any certificates. However, it does require the exchange of usernames and passwords with the administrator of the trusted provider, which might or might not compromise the security of the trusted relationship.

If your trusted provider is another Identity Server, you can use any of these methods, as long as your Identity Server and the trusted Identity Server use the same method. If you are setting up a trusted relationship with a third-party provider, you need to select a method supported by that provider.

For configuration information, see the following sections:

- Section 7.5.1, “Configuring Communication Security for Liberty and SAML 1.1,” on page 201
- Section 7.5.2, “Configuring Communication Security for a SAML 2.0 Identity Provider,” on page 201
- Section 7.5.3, “Configuring Communication Security for a SAML 2.0 Service Provider,” on page 203

7.5.1 Configuring Communication Security for Liberty and SAML 1.1

Liberty and SAML 1.1 have the same security options for the SOAP back channel for both identity and service providers. You cannot configure the trust relationship of the SOAP back channel for the Identity Server and its Embedded Service Providers.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *[Protocol]*.

   For the protocol, select either Liberty or SAML 1.1.

2. Click the name of a provider.

3. On the Trust page, fill in the following field:

   Name: Specify the display name for this trusted provider. The default name is the name you entered when creating the trusted provider.

   For an Embedded Service Provider, the *Name* option is the only available option on the Trust page.

   The *Security* section specifies how to validate messages received from trusted providers over the SOAP back channel. Both the identity provider and the service provider in the trusted relationship must be configured to use the same security method.

4. Select one of the following security methods:

   Message Signing: Relies upon message signing using a digital signature.

   Mutual SSL: Specifies that this trusted provider provides a digital certificate (mutual SSL) when it sends a SOAP message.

   SSL communication requires only the client to trust the server. For mutual SSL, the server must also trust the client. For the client to trust the server, the server's certificate authority (CA) certificate must be imported into the client trust store. For the server to trust the client, the client's CA certificate must be imported into the server trust store.

   Basic Authentication: Specifies standard header-based authentication. This method assumes that a name and password for authentication are sent and received over the SOAP back channel.

   - **Send:** The name and password to be sent for authentication to the trusted partner. The partner expects this password for all SOAP back-channel requests, which means that the name and password must be agreed upon.
   - **Verify:** The name and password used to verify data that the trusted provider sends.

5. Click *OK* twice.

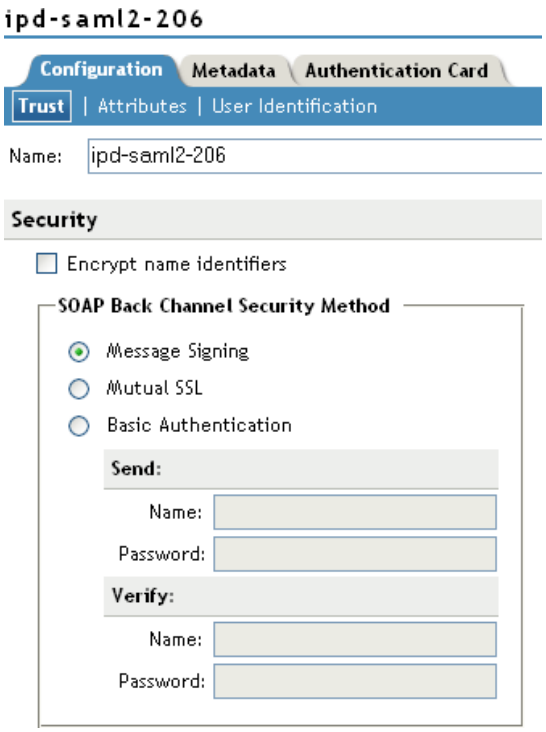
6. Update the Identity Server.

7.5.2 Configuring Communication Security for a SAML 2.0 Identity Provider

The security settings control the direct communication between the Identity Server and the identity provider across the SOAP back channel.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 2.0*.

2 Click the name of an identity provider.



3 On the Trust page, fill in the following fields:

Name: Specify the display name for this trusted provider. The default name is the name you entered when creating the trusted provider.

The *Security* section specifies how to validate messages received from trusted providers over the SOAP back channel. Both the identity provider and the service provider in the trusted relationship must be configured to use the same security method.

Encrypt name identifiers: Specifies whether you want the name identifiers encrypted on the wire.

Select one of the following security methods:

- **Message Signing:** Relies upon message signing using a digital signature.
- **Mutual SSL:** Specifies that this trusted provider provides a digital certificate (mutual SSL) when it sends a SOAP message.

  SSL communication requires only the client to trust the server. For mutual SSL, the server must also trust the client. For the client to trust the server, the server's certificate authority (CA) certificate must be imported into the client trust store. For the server to trust the client, the client's CA certificate must be imported into the server trust store.
- **Basic Authentication:** Specifies standard header-based authentication. This method assumes that a name and password for authentication are sent and received over the SOAP back channel.

  Send: The name and password to be sent for authentication to the trusted partner. The partner expects this password for all SOAP back-channel requests, which means that the name and password must be agreed upon.

  Verify: The name and password used to verify data that the trusted provider sends.

4 Click *OK* twice.

5 Update the Identity Server.

7.5.3 Configuring Communication Security for a SAML 2.0 Service Provider

The security settings control the direct communication between the Identity Server and the service provider across the SOAP back channel.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 2.0*.

2 Click the name of a service provider.

3 On the Trust page, fill in the following fields:

Name: Specify the display name for this trusted provider. The default name is the name you entered when creating the trusted provider.

The *Security* section specifies how to validate messages received from trusted providers over the SOAP back channel. Both the identity provider and the service provider in the trusted relationship must be configured to use the same security method.

Encrypt assertions: Specifies whether you want the assertions encrypted on the wire.

Encrypt name identifiers: Specifies whether you want the name identifiers encrypted on the wire.

Select one of the following security methods:

- **Message Signing:** Relies upon message signing using a digital signature.
- **Mutual SSL:** Specifies that this trusted provider provides a digital certificate (mutual SSL) when it sends a SOAP message.

  SSL communication requires only the client to trust the server. For mutual SSL, the server must also trust the client. For the client to trust the server, the server's certificate authority (CA) certificate must be imported into the client trust store. For the server to trust the client, the client's CA certificate must be imported into the server trust store.
- **Basic Authentication:** Specifies standard header-based authentication. This method assumes that a name and password for authentication are sent and received over the SOAP back channel.

  Send: The name and password to be sent for authentication to the trusted partner. The partner expects this password for all SOAP back-channel requests, which means that the name and password must be agreed upon.

  Verify: The name and password used to verify data that the trusted provider sends.

4 Click *OK* twice.

5 Update the Identity Server.

7.6 Selecting Attributes for a Trusted Provider

You can select attributes that an identity provider sends in an authentication request or that a service provider receives in an authentication response. The attributes are selected from an attribute set, which you can create or select from a list of already defined sets (see Section 6.1, "Configuring Attribute Sets," on page 177).

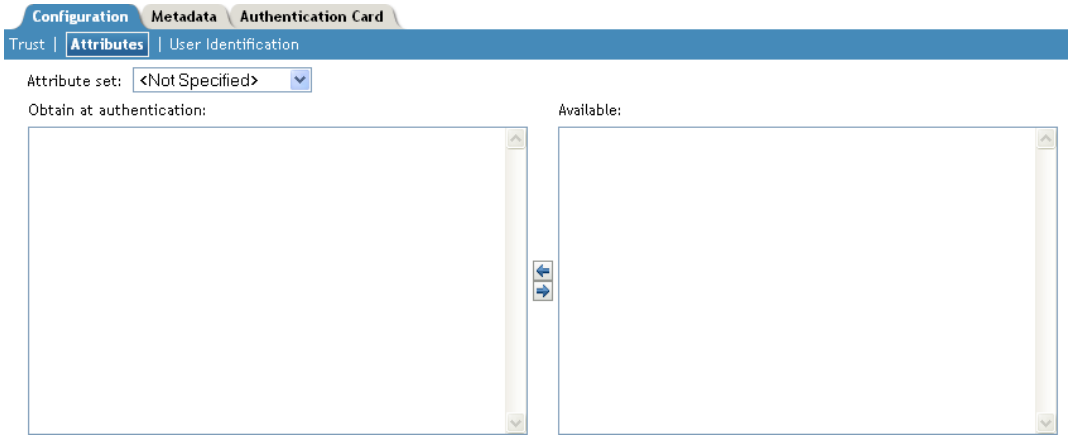
For best performance, you should configure the trusted providers to use attribute sets, especially for attributes that have static values such as a user's e-mail address, employee ID, or phone number. It reduces the traffic between the provider and the LDAP server, because the attribute information can be gathered in one request at authentication rather than in a separate request for each attribute when a policy or protected resource needs the attribute information.

- Section 7.6.1, "Configuring the Attributes Obtained at Authentication," on page 204
- Section 7.6.2, "Configuring the Attributes Sent with Authentication," on page 205
- Section 7.6.3, "Sending Attributes to the Embedded Service Provider," on page 206

7.6.1 Configuring the Attributes Obtained at Authentication

When the Identity Server creates its request to send to the identity provider, it uses the attributes that you have selected. The request asks the identity provider to provide values for these attributes. You can then use these attributes to create policies, to match user accounts, or if you allow provisioning, to create a user account on the service provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *[Protocol]* > *[Identity Provider]* > *Attributes*.



2 (Conditional) To create an attribute set, select *New Attribute Set* from the *Attribute Set* drop-down menu.

An attribute set is a group of attributes that can be exchanged with the trusted provider. For example, you can specify that the local attribute of any attribute in the Liberty profile (such as Informal Name) matches the remote attribute specified at the service provider.

2a Specify a set name, then click *Next*.

2b On the Define Attributes page, click *New*.

2c Select a local attribute.

2d Optionally, provide the name of the remote attribute and a namespace.

2e Click *OK*.

For more information on this process, see Section 6.1, "Configuring Attribute Sets," on page 177.

2f To add other attributes to the set, repeat Step 2b through Step 2e.

2g Click *Finish*.

3 Select an attribute set

4 Select attributes from the *Available* list, and move them to the left side of the page.

The attributes that you move to the left side of the page are the attributes you want to be obtained during authentication.

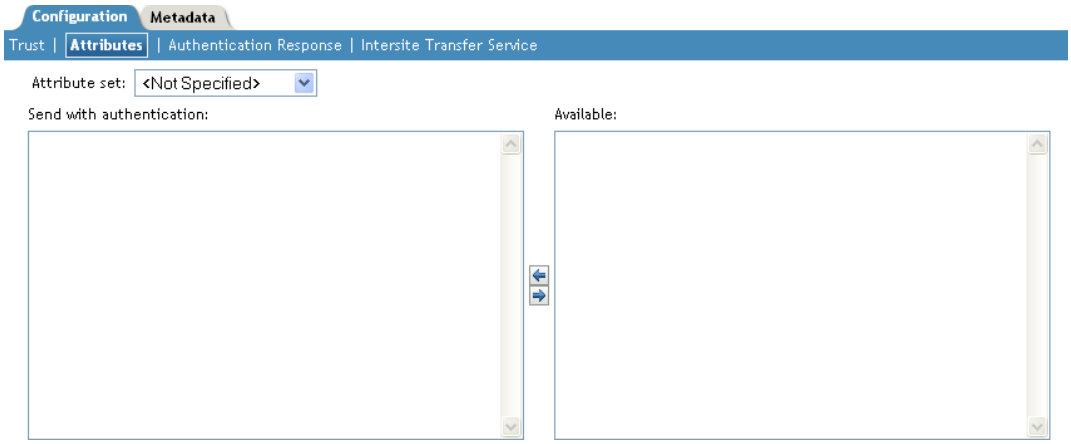
5 Click *OK* twice.

6 Update the Identity Server.

7.6.2 Configuring the Attributes Sent with Authentication

When the Identity Server creates its response for the service provider, it uses the attributes listed on the Attributes page. The response needs to contain the attributes that the service provider requires. If you do not own the service provider, you need to contact the administrator of the service provider and negotiate which attributes you need to send in the response. The service provider can then use these attributes to identify the user, to create policies, to match user accounts, or if it allows provisioning, to create a user accounts on the service provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *[Protocol]* > *[Service Provider]* > *Attributes*.



2 (Conditional) To create an attribute set, select *New Attribute Set* from the *Attribute Set* drop-down menu.

An attribute set is a group of attributes that can be exchanged with the trusted provider. For example, you can specify that the local attribute of any attribute in the Liberty profile (such as Informal Name) matches the remote attribute specified at the service provider.

2a Specify a set name, then click *Next*.

2b On the Define Attributes page, click *New*.

2c Select a local attribute.

2d Optionally, you can provide the name of the remote attribute and a namespace.

2e Click *OK*.

For more information on this process, see Section 6.1, “Configuring Attribute Sets,” on page 177.

2f To add other attributes to the set, repeat Step 2b through Step 2e.

2g Click *Finish*.

3 Select an attribute set

4 Select attributes from the *Available* list, and move them to the left side of the page.

The left side of the page lists the attributes that you want sent in an assertion to the service provider.

5 Click *OK* twice.

6 Update the Identity Server.

7.6.3 Sending Attributes to the Embedded Service Provider

You can configure the Embedded Service Provider (ESP) of the Access Gateway to receive attributes when the user authenticates. LDAP traffic is reduced and performance is improved when the required LDAP attribute values are retrieved during authentication. This improvement is easily seen when you have many users and you have configured Identity Injection or Authorization policies to protect resources and these policies use LDAP attributes or Identity Server roles.

When the authentication process does not gather the LDAP attribute values, each user access can generate a new LDAP query, depending upon how the user accesses the resources and how the policies are defined. However, if the LDAP values are gathered at authentication, one LDAP query can retrieve all the needed values for the user.

1 In the Administration Console, click *Devices* > *Identity Servers* > Shared Settings.

2 On the Attributes page, click *New*, specify a name, then click *Next*.

3 For each attribute you need to add because it is used in a policy:

3a Click *New*.

3b In the *Local attribute* drop-down list, scroll to LDAP Attribute section, then select the attribute.

3c Click *OK*.

The other fields do not need to be configured.

4 If you use Identity Server roles in your policies, click *New*, select the All Roles attribute, then click *OK*.

5 Click *Finish*.

This saves the attribute set.

6 Click *Servers* > *Edit* > *Liberty*.

7 Click the name of the Embedded Service Provider.

If the Embedded Service Provider is part of a cluster of Access Gateways, the default name is the cluster name. If the Access Gateway is not part of a cluster, the default name is the IP address of the Access Gateway.

8 Click *Attributes*.

9 For the attribute set, select the set you created for the Embedded Service Provider.

10 Select attributes from the *Available* list, then move them to the left side of the page.

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

7.7 Managing Metadata

The Liberty, SAML 1.1, and SAML 2.0 protocols contain pages for viewing and reimporting the metadata of the trusted providers. Only the SAML 1.1 protocol allows you to edit the metadata.

7.7.1 Viewing and Reimporting a Trusted Provider's Metadata

You might need to reimport a trusted provider's metadata if you learn that it has changed. The metadata changes when you change the provider to use HTTPS rather than HTTP and when you change the certificate that it is using for SSL. The steps for reimporting the metadata are similar for Liberty and SAML protocols.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *[Protocol]*.
2. Click the trusted provider, then click the *Metadata* tab.

   This page displays the current metadata the trusted provider is using.
3. To reimport the metadata:
   - **3a** Copy the URL in the providerID field (Liberty) or the entityID (SAML).
   - **3b** (SAML 1.1) Paste the URL to a file, click *Authentication Card*, copy the *Login URL* to the file, then click *Metadata*.
   - **3c** Click *Reimport*.
   - **3d** Follow the prompts to import the metadata.

     For the metadata URL, paste in the value you copied.

     If your Administration Console is installed with your Identity Server, you need to change the protocol from HTTPS to HTTP and the port from 8443 to 8080.
4. Confirm metadata certificates, then click *Finish*, or for an identity provider, click *Next*.
5. (Identity Provider) Configure the card, then click *Finish*.

   For SAML 1.1, copy the value you saved into the *Login URL*.
6. Update the Identity Server.

7.7.2 Viewing Trusted Provider Certificates

You can review and confirm the certificate information for identity and service providers.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *[Protocol]* > *[Name of Provider]* > *Metadata* > *Certificates*.
2. View the following information is displayed for the certificates:

   Subject: The subject name assigned to the certificate.

   Validity: The first date the certificate was valid, and the date the certificate expires.

Issuer DN: The distinguished name of the Certificate Authority (CA) that created the certificate.

Algorithm: The name of the algorithm that was used to create the certificate.

Serial Number: The serial number that the CA assigned to the certificate.

3 Click *OK* if you are viewing the information, or click *Next* or *Finish* if you are creating a provider.

7.7.3 Editing a SAML 1.1 Identity Provider's Metadata

Access Manager allows you to import metadata for SAML 1.1 providers. However, metadata for SAML 1.1 might not be available for some trusted providers, so you can enter metadata manually. The page for this is available if you clicked the *Manual Entry* option when you created the trusted provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 1.1* > *[Identity Provider]* > *Metadata*.

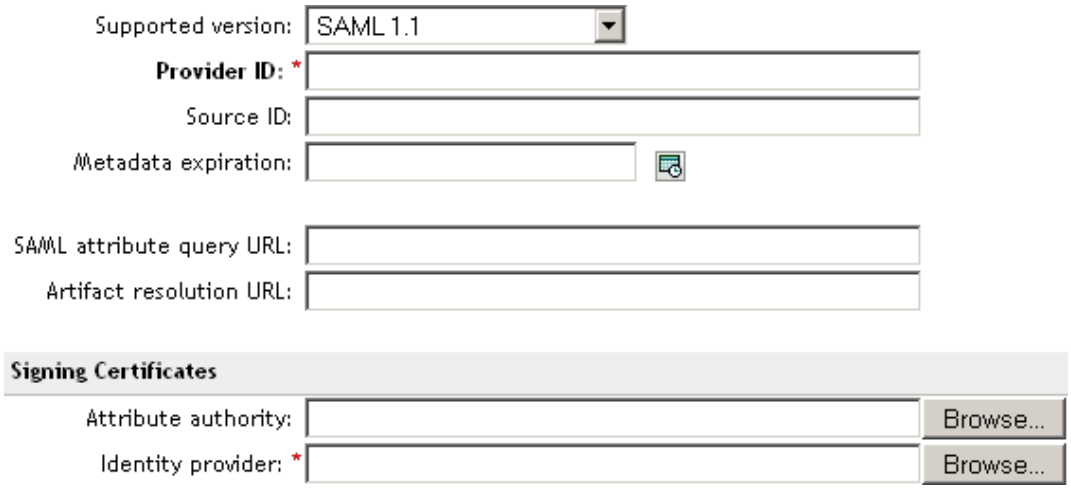
You can reimport the metadata (see Step 2) or edit it (see Step 4).

2 To reimport the metadata from a URL or text, click *Reimport* on the View page.

The system displays the Create Trusted Identity Provider Wizard that lets you obtain the metadata. Follow the on-screen instructions to complete the steps in the wizard.

3 Select either *Metadata URL* or *Metadata Text*, then fill in the field for the metadata.

4 To edit the metadata manually, click *Edit*.



5 Fill in the following fields as necessary:

Supported Version: Specifies the version of SAML that you want to use. You can select SAML 1.0, SAML 1.1, or both SAML 1.0 and SAML 1.1.

Provider ID: (Required) The SAML 1.1 metadata unique identifier for the provider. For example, `https://<dns>:8443/nidp/saml/metadata`. Replace <*dns*> with the DNS name of the provider.

In the metadata, this is the entityID value.

Source ID: The SAML Source ID for the trusted provider. The Source ID is a 20-byte value that is used as part of the Browser/Artifact profile. It allows the receiving site to determine the source of received SAML artifacts. If none is specified, the Source ID is auto-generated by using a SHA-1 hash of the site provider ID.

Metadata expiration: The date upon which the metadata is no longer valid.

SAML attribute query URL: The URL location where an attribute query is to be sent to the partner. The attribute query requests a set of attributes associated with a specific object. A successful response contains assertions that contain attribute statements about the subject. A SAML 1.1 provider might use the base URL, followed by /saml/soap. For example, `https://<dns>:8443/nidp/saml/soap`. Replace <*dns*> with the DNS name of the provider.

In the metadata, this URL value is found in the AttributeService section of the metadata.

Artifact resolution URL: The URL location where artifact resolution queries are sent. A SAML artifact is included in the URL query string. The target URL on the destination site the user wants to access is also included on the query string. A SAML 1.1 provider might use the base URL, followed by /saml/soap. For example, `https://<dns>:8443/nidp/saml/soap`. Replace <*dns*> with the DNS name of the provider.

In the metadata, this URL value is found in the ArtifactResolutionService section of the metadata.

6 To specify signing certificate settings, fill in the following fields:

Attribute authority: Specifies the signing certificate of the partner SAML 1.1 attribute authority. The attribute authority relies on the identity provider to provide it with authentication information so that it can retrieve attributes for the appropriate entity or user. The attribute authority must know that the entity requesting the attribute has been authenticated to the system.

Identity provider: (Required) Appears if you are editing identity provider metadata. This field specifies the signing certificate of the partner SAML 1.1 identity provider. It is the certificate the partner uses to sign authentication assertions.

7 Click *OK*.

8 On the Identity Servers page, click *Update All* to update the configuration.

7.7.4 Editing a SAML 1.1 Service Provider's Metadata

Access Manager allows you to obtain metadata for SAML 1.1 providers. However, metadata for SAML 1.1 might not be available for some trusted providers, so you can enter the metadata manually. The page for this is available if you clicked the *Manual Entry* option when you created the trusted provider.

For conceptual information about how Access Manager uses SAML, see Appendix B, "Understanding How Access Manager Uses SAML," on page 369.

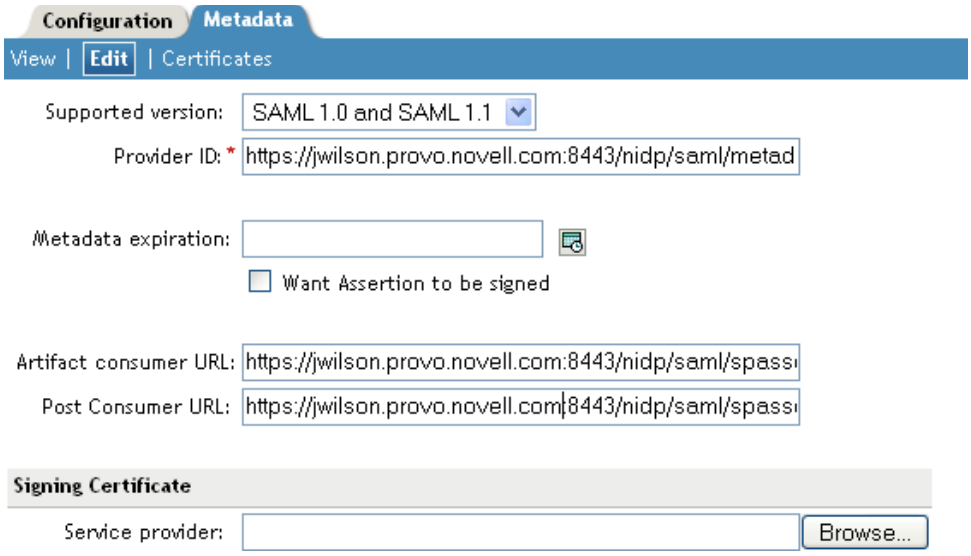
1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 1.1* > *[Service Provider]* > *Metadata*.

You can reimport the metadata (see Step 2) or edit it (see Step 3).

2 To reimport the metadata, click *Reimport* on the View page.

Follow the on-screen instructions to complete the steps in the wizard.

3 To edit the metadata manually, click *Edit*.



4 Fill in the following fields:

Supported Version: Specifies which version of SAML that you want to use. You can select SAML 1.0, SAML 1.1, or both SAML 1.0 and SAML 1.1.

Provider ID: (Required) Specifies the SAML 1.1 metadata unique identifier for the provider. For example, `https://<dns>:8443/nidp/saml/metadata`. Replace <*dns*> with the DNS name of the provider.

In the metadata, this is the entityID value.

Metadata expiration: Specifies the date upon which the metadata is no longer valid.

Want assertion to be signed: Specifies that authentication assertions from the trusted provider must be signed.

Artifact consumer URL: Specifies where the partner receives incoming SAML artifacts. For example, `https://<dns>:8443/nidp/saml/spassertion_consumer`. Replace <*dns*> with the DNS name of the provider.

In the metadata, this URL value is found in the AssertionConsumerService section of the metadata.

Post consumer URL: Specifies where the partner receives incoming SAML POST data. For example, `https://<dns>:8443/nidp/saml/spassertion_consumer`. Replace <*dns*> with the DNS name of the provider.

In the metadata, this URL value is found in the AssertionConsumerService section of the metadata.

Service Provider: Specifies the public key certificate used to sign SAML data. You can browse to locate the service provider certificate.

5 Click *Finish*.

7.8 Configuring an Authentication Request for an Identity Provider

When you are configuring the Identity Server to trust an identity provider and to use that identity provider for authentication, you can specify the conditions under which the Identity Server accepts the authentication credentials of the identity provider. The authentication request contains these conditions.

The Liberty and SAML 2.0 protocols have slightly different options for configuring an authentication request.

- Section 7.8.1, "Configuring a Liberty Authentication Request," on page 211
- Section 7.8.2, "Configuring a SAML 2.0 Authentication Request," on page 213

7.8.1 Configuring a Liberty Authentication Request

You can configure how the Identity Server creates an authentication request for a trusted identity provider. When users authenticate, they can be given the option to federate their account identities with the preferred identity provider. This process creates an account association between the identity provider and service provider that enables single sign-on and single log-out.

The authentication request specifies how you want the identity provider to handle the authentication process so that it meets the security needs of the Identity Server.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *[Identity Provider]* > *Authentication Card* > *Authentication Request*.
2. Configure the federation options:

   Allow Federation: Determines whether federation is allowed. The federation options that control when and how federation occurs can only be configured if the identity provider has been configured to allow federation.

   - **After authentication:** Specifies that the federation request can be sent after the user has authenticated (logged in) to the service provider. When you set only this option, users must log in locally, then they can federate by using the *Federate* option on the card in the Login page of the Access Manager User Portal. Because the user is required to authenticate locally, you do not need to set up user identification.
   - **During authentication:** Specifies whether federation can occur when the user selects the authentication card of the identity provider. Typically, a user is not authenticated at the service provider when this selection is made. When the identity provider sends a response to the service provider, the user needs to be identified on the service provider to complete the federation. If you enable this option, make sure you configure a user identification method. See Section 11.1.1, "Selecting a User Identification Method for Liberty or SAML 2.0," on page 281.
3. Select one of the following options for the *Requested By* option:

   Do not specify: Specifies that the identity provider can send any type of authentication to satisfy a service provider's request, and instructs a service provider to not send a request for a specific authentication type or contract.

   Use Types: Specifies that authentication types should be used.

Select the types from the *Available types* field to specify which type to use for authentication between trusted service providers and identity providers. Standard types include Name/Password, Secure Name/Password, X509, Token, and so on.

Use Contracts: Specifies that authentication contracts should be used.

Select the contract from the *Available contracts* list. For a contract to appear in the *Available contracts* list, the contract must have the *Satisfiable by External Provider* option enabled. To use the contract for federated authentication, the contract's URI must be the same on the identity provider and the service provider. For information about contract options, see Section 3.4, "Configuring Authentication Contracts," on page 125.

Most third-party identity providers do not use contracts.

4 Configure the options:

Response protocol binding: Select *Artifact* or *Post* or *None*. Artifact and Post are the two methods for transmitting assertions between the authenticating system and the target system.

If you select *None*, you are letting the identity provider determine the binding.

Identity Provider proxy redirects: Specifies whether the trusted identity provider can proxy the authentication request to another identity provider. A value of *None* specifies that the trusted identity provider cannot redirect an authentication request. Values 1-5 determine the number of times the request can be proxied. Select *Configured on IDP* to let the trusted identity provider decide how many times the request can be proxied.

Force authentication at Identity Provider: Specifies that the trusted identity provider must prompt users for authentication, even if they are already logged in.

Use automatic introduction: Attempts single sign-on to this trusted identity provider by automatically sending a passive authentication request to the identity provider. (A passive requests does not prompt for credentials.) The identity provider sends one of the following authentication responses:

- **When the federated user is authenticated at the identity provider:** The identity provider returns an authentication response indicating that the user is authenticated. The user gains access to the service provider without entering credentials (single sign-on).
- **When the federated user is not authenticated at the identity provider:** The identity provider returns an authentication response indicating that the user is not logged in. The user can then select a card for authentication, including the card for the identity provider. If the user selects the identity provider card, an authentication request is sent to the identity provider. If the credentials are valid, the user is also authenticated to the service provider.

IMPORTANT: Enable the *Use automatic introduction* option only when you are confident the identity provider will be up. If the server is down and does not respond to the authentication request, the user gets a page-cannot-be-displayed error. Local authentication is disabled because the browser is never redirected to the login page.

This option should be enabled only when you know the identity provider is available 99.999% of the time or when the service provider is dependent upon this identity provider for authentication.

---

5 Click *OK* twice, then update the Identity Server.

7.8.2 Configuring a SAML 2.0 Authentication Request

You can configure how an authentication request is federated. When users authenticate to a service provider, they can be given the option to federate their account identities with the preferred identity provider. This process creates an account association between the identity provider and service provider that enables single sign-on and single log-out.

The authentication request specifies how you want the identity provider to handle the authentication process so that it meets the security needs of the Identity Server.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 2.0* > *[Identity Provider]* > *Authentication Card* > *Authentication Request.*
2. Configure the name identifier format:

   Persistent: Specifies that a persistent identifier, which is written to the directory and remains intact between sessions, can be sent.

   - **After authentication:** Specifies that the persistent identifier can be sent after the user has authenticated (logged in) to the service provider. When you set only this option, users must log in locally. Because the user is required to authenticate locally, you do not need to set up user identification.
   - **During authentication:** Specifies that the persistent identifier can be sent when the user selects the authentication card of the identity provider. Typically, a user is not authenticated at the service provider when this selection is made. When the identity provider sends a response to the service provider, the user needs to be identified on the service provider. If you enable this option, make sure you configure a user identification method. See Section 11.1.1, “Selecting a User Identification Method for Liberty or SAML 2.0,” on page 281.

   Transient: Specifies that a transient identifier, which expires between sessions, can be sent.

   Unspecified: Allows either a persistent or a transient identifier to be sent.
3. Select one of the following options for the *Requested By* option:

   Do not specify: Specifies that the identity provider can send any type of authentication to satisfy a service provider’s request, and instructs a service provider to not send a request for a specific authentication type or contract.

   Use Types: Specifies that authentication types should be used.

   Select the type of comparison (for more information, see “Understanding Comparison Contexts” on page 215):

   - **Exact:** Indicates that the class or type specified in the authentication statement must be an exact match to at least one contract.
   - **Minimum:** Indicates that the contract must be as strong as the class or type specified in the authentication statement.
   - **Better:** Indicates the contract that must be stronger than the class or type specified in the authentication statement.
   - **Maximum:** Indicates that contract must as strong as possible without exceeding the strength of at least one of the authentication contexts specified.

   Select the types from the *Available types* field to specify which type to use for authentication between trusted service providers and identity providers. Standard types include Name/Password, X.509, Token, and so on.

Use Contracts: Specifies that authentication contracts should be used.

Select the type of comparison (for more information, see "Understanding Comparison Contexts" on page 215):

- **Exact:** Indicates that the class or type specified in the authentication statement must be an exact match to at least one contract.
- **Minimum:** Indicates that the contract must be as strong as the class or type specified in the authentication statement.
- **Better:** Indicates the contract that must be stronger than the class or type specified in the authentication statement.
- **Maximum:** Indicates that contract must as strong as possible without exceeding the strength of at least one of the authentication contexts specified.

Select the contract from the *Available contracts* list. For a contract to appear in the *Available contracts* list, the contract must have the *Satisfiable by External Provider* option enabled. To use the contract for federated authentication, the contract's URI must be the same on the identity provider and the service provider. For information about contract options, see Section 3.4, "Configuring Authentication Contracts," on page 125.

Most third-party identity providers do not support contracts.

4 Configure the options:

Response protocol binding: Select *Artifact* or *Post* or *None*. Artifact and Post are the two methods for transmitting assertions between the authenticating system and the target system.

If you select *None*, you are letting the identity provider determine the binding.

Allowable IDP proxy indirections: Specifies whether the trusted identity provider can proxy the authentication request to another identity provider. A value of *None* specifies that the trusted identity provider cannot redirect an authentication request. Values 1-5 determine the number of times the request can be proxied. Select *Let IDP Decide* to let the trusted identity provider decide how many times the request can be proxied

Force authentication at Identity Provider: Specifies that the trusted identity provider must prompt users for authentication, even if they are already logged in.

Use automatic introduction: Attempts single sign-on to this trusted identity provider by automatically sending a passive authentication request to the identity provider. (A passive requests does not prompt for credentials.) The identity provider sends one of the following authentication responses:

- **When the federated user is authenticated at the identity provider:** The identity provider returns an authentication response indicating that the user is authenticated. The user gains access to the service provider without entering credentials (single sign-on).
- **When the federated user is not authenticated at the identity provider:** The identity provider returns an authentication response indicating that the user is not logged in. The user can then select a card for authentication, including the card for the identity provider. If the user selects the identity provider card, an authentication request is sent to the identity provider. If the credentials are valid, the user is also authenticated to the service provider.

IMPORTANT: Enable the *Use automatic introduction* option only when you are confident the identity provider will be up. If the server is down and does not respond to the authentication request, the user gets a page-cannot-be-displayed error. Local authentication is disabled because the browser is never redirected to the login page.

This option should be enabled only when you know the identity provider is available 99.999% of the time or when the service provider is dependent upon this identity provider for authentication.

5 Click *OK* twice, then update the Identity Server.

Understanding Comparison Contexts

When a service provider makes a request for an identity provider to authenticate a user, the authentication request can contain a class or type and a comparison context. The identity provider uses these to determine which authentication procedure to execute. There are four comparison contexts:

- **Exact:** Indicates that the class or type specified in the authentication statement must be an exact match to at least one contract.

  For example, when the comparison context is set to exact, the identity provider uses the URI in the request to find an authentication procedure. If an exact URI match is found, the user is prompted for the appropriate credentials. If an exact match is not found, the user is denied access.

- **Better:** Indicates the contract that must be stronger than the class or type specified in the authentication statement.

  If the identity provider is a Novell Identity Server, the Identity Server first finds the specified class or type and its assigned authentication level. It then uses this information to find a contract that matches the conditions. For example if the authentication level is set to 1 for the class or type, the identity provider looks for a contract with an authentication level that is higher than 1. If one is found, the user is prompted for the appropriate credentials. If more than one is found, the user is presented with the matching cards and is allowed to select the contract. If a match is not found, the user is denied access.

- **Minimum:** Indicates that the contract must be as strong as the class or type specified in the authentication statement.

  If the identity provider is a Novell Identity Server, the Identity Server first finds the specified class or type and its assigned authentication level. It then uses this information to find a contract that matches the conditions. For example if the authentication level is set to 1 for the class or type, the identity provider looks for a contract with an authentication level of 1 or higher. If one is found, the user is prompted for the appropriate credentials. If more than one is found, the user is presented with the matching cards and is allowed to select the contract. If a match is not found, the user is denied access.

- **Maximum:** Indicates that contract must as strong as possible without exceeding the strength of at least one of the authentication contexts specified.

  If the identity provider is a Novell Identity Server, the Identity Server first finds the specified classes or types and their assigned authentication levels. It then uses this information to find a contract that matches the conditions. For example if the authentication level is set to 1 for some types and 3 for other types, the identity provider looks for contracts with an authentication level of 3. If a match or matches are found, the user is presented with the appropriate login prompts. If there are no contracts defined with a authentication level of 3, the identity provider looks for a match with an authentication level of 2, or if necessary, level 1. It cannot search below the lowest level of class in the authentication request or higher than the highest level of a class in the authentication request.

When you configure the authentication request, you specify the comparison context for a type or a contract.

7.9 Configuring an Authentication Response for a Service Provider

The Liberty and SAML 2.0 protocols support slightly different options for configuring how you want the Identity Server to respond to an authentication request from a service provider. The SAML 1.1 protocol does not support sending an authentication request. However, you can configure an Intersite Transfer Service (see Section 7.11, "Using the Intersite Transfer Service," on page 221) to trigger a response from the Identity Server.

When the Identity Server receives an authentication request from a trusted service provider, the request contains the conditions that the Identity Server needs to fulfill. The Authentication Response page allows you to configure how you want the Identity Server to fulfill the binding and name identifier conditions of the request, or for SAML 1.1, respond to the Intersite Transfer Service. For configuration information, see one of the following:

- Section 7.9.1, "Configuring the Liberty Authentication Response," on page 216
- Section 7.9.2, "Configuring the SAML 2.0 Authentication Response," on page 217
- Section 7.9.3, "Configuring the SAML 1.1 Authentication Response," on page 219

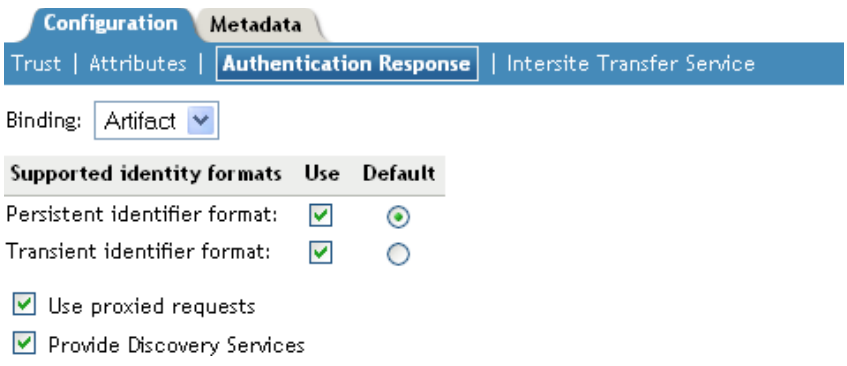
The Defaults page allows you to specify which contract is used when the authentication request specifies a class or type rather than a contract. For more information, see Section 3.5, "Specifying Authentication Defaults," on page 131.

When the service provider sends an authentication request that specifies a specific contract, you need to ensure that the Identity Server has a the contract matches the expected URI. For information on how to configure such a contract, see Section 3.5.2, "Creating a Contract for a Specific Authentication Type," on page 133.

7.9.1 Configuring the Liberty Authentication Response

After you create a trusted service provider, you can configure how your Identity Server responds to authentication requests from the service provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *[Service Provider]* > *Authentication Response*.



2 Select the binding method.

If the request from the service provider does not specify a response binding, you need to specify a binding method to use in the response. Select *Artifact* to provide an increased level of security by using a back-channel means of communication between the two servers. Select *Post* to use HTTP redirection for the communication channel between the two servers. If you select *Post*, you might want to require the signing of the authentication requests. See Section 7.2.1, "Configuring the General Identity Provider Options," on page 190.

3 Specify the identity formats that the Identity Server can send in its response. Select the *Use* box to choose one or more of the following:

- **Persistent Identifier Format:** Specifies that a persistent identifier, which is written to the directory and remains intact between sessions, can be sent.
- **Transient Identifier Format:** Specifies that a transient identifier, which expires between sessions, can be sent.

If the request from the service provider requests a format that is not enabled, the user cannot authenticate.

4 Use the *Default* button to specify whether a persistent or transient identifier is sent when the request from the service provider does not specify a format.

5 To specify that this Identity Server must authenticate the user, disable the *Use proxied requests* option. When the option is disabled and the Identity Server cannot authenticate the user, the user is denied access.

When this option is enabled, the Identity Server checks to see if other identity providers can satisfy the request. If one or more can, the user is allowed to select which identity provider performs the authentication. If a proxied identity provider performs the authentication, it sends the response to the Identity Server. The Identity Server then sends the response to the service provider.

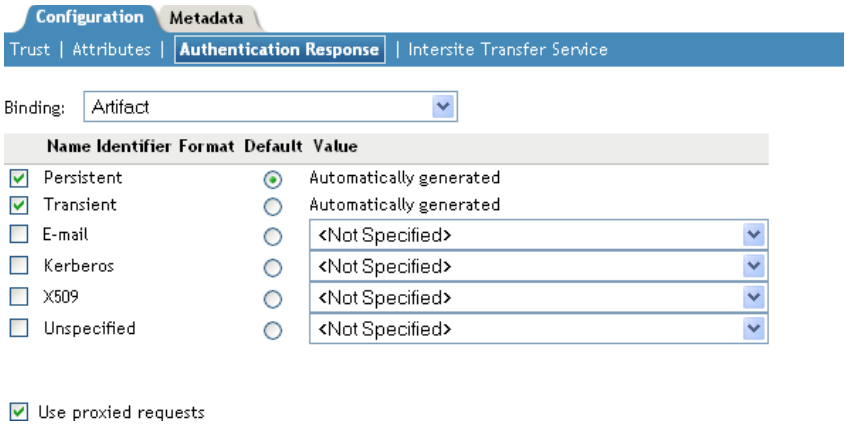
6 Enable the *Provide Discovery Services* option if you want to allow the service provider to query the Identity Server for a list of its Web Services. For example, when the option is enabled, the service provider can determine whether the Web Services Framework is enabled and which Web Service Provider profiles are enabled.

7 Click *OK* twice, then update the Identity Server.

7.9.2 Configuring the SAML 2.0 Authentication Response

After you create a trusted service provider, you can configure how your Identity Server responds to authentication requests from the service provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 2.0* > *[Service Provider]* > *Authentication Response*.



2 Select the binding method.

If the request from the service provider does not specify a response binding, you need to specify a binding method to use in the response. Select *Artifact* to provide an increased level of security by using a back-channel means of communication between the two servers. Select *Post* to use HTTP redirection for the communication channel between the two servers. If you select *Post*, you might want to require the signing of the authentication requests. See Section 7.2.1, "Configuring the General Identity Provider Options," on page 190.

3 Specify the identity formats that the Identity Server can send in its response. Select the box to choose one or more of the following:

- **Persistent:** Specifies that a persistent identifier, which is written to the directory and remains intact between sessions, can be sent.
- **Transient:** Specifies that a transient identifier, which expires between sessions, can be sent.
- **E-mail:** Specifies that an e-mail attribute can be used as the identifier.
- **Kerberos:** Specifies that a Kerberos token can be used as the identifier.
- **X509:** Specifies that an X.509 certificate can be used as the identifier.
- **Unspecified:** Specifies that an unspecified format can be used and any value can be used. The service provider and the identity provider need to agree on the value that is placed in this identifier.

4 Use the *Default* button to select the name identifier that the Identity Server should send if the service provider does not specify a format.

If you select E-mail, Kerberos, x509, or unspecified as the default format, you should also select a value. See Step 5.

IMPORTANT: If you have configured the identity provider to allow a user matching expression to fail and still allow authentication by selecting the *Do nothing* option, you need to select *Transient identifier format* as the default value. Otherwise the users who fail the matching expression are denied access. To view the identity provider configuration, see "Defining User Identification for Liberty and SAML 2.0" on page 281.

---

5 Specify the value for the name identifier.

The persistent and transient formats are generated automatically. For the others, you can select an attribute. The available attributes depend upon the attributes that you have selected to send with authentication (see Section 7.6.1, "Configuring the Attributes Obtained at Authentication," on page 204). If you do not select a value for the E-mail, Kerberos, X509, or Unspecified format, a unique value is automatically generated.

6 To specify that this Identity Server must authenticate the user, disable the *Use proxied requests* option. When the option is disabled and the Identity Server cannot authenticate the user, the user is denied access.

When this option is enabled, the Identity Server checks to see if other identity providers can satisfy the request. If one or more can, the user is allowed to select which identity provider performs the authentication. If a proxied identity provider performs the authentication, it sends the response to the Identity Server. The Identity Server then sends the response to the service provider.

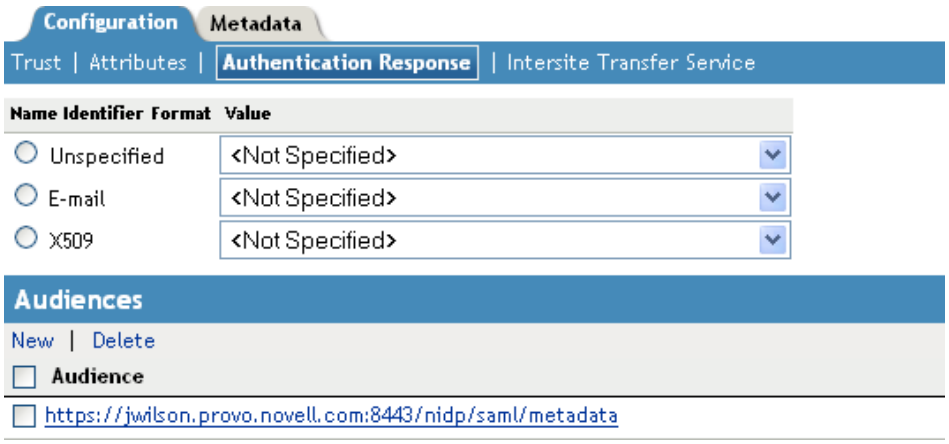
7 Click *OK* twice, then update the Identity Server.

7.9.3 Configuring the SAML 1.1 Authentication Response

You can specify the name identifier and its format when the Identity Server sends an authentication response. You can also restrict the use of the assertion.

When an identity provider sends an assertion, the assertion can be restricted to an intended audience. The intended audience is defined to be any abstract URI in SAML 1.1. The URL reference can also identify a document that describes the terms and conditions of audience membership.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 1.1* > *[Service Provider]* > *Authentication Response*.



2 To specify a name identifier format, select one of the following:

- **E-mail:** Specifies that an e-mail attribute can be used as the identifier.
- **X509:** Specifies that an X.509 certificate can be used as the identifier.
- **Unspecified:** Specifies that an unspecified format can be used and any value can be used. The service provider and the identity provider need to agree on what value is placed in this identifier.

3 To specify the format of the name identifier, select an attribute.

The available attributes depend upon the attributes that you have selected to send with authentication (see the Attributes page for the service provider).

4 To configure an audience, click *New*.

5 Specify the *SAML Audience URL* value.

The Provider ID, which can be used for the audience, is displayed on the Edit page for the metadata.

6 Click *OK* twice, then update the Identity Server.

7.10 Managing the Authentication Card of an Identity Provider

7.10.1 Modifying the Authentication Card for Liberty or SAML 2.0

When you create an identity provider, you must also configure an authentication card. After it is created, you can modify it.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *[Protocol]* > *[Identity Provider]* > *Authentication Card.*

2 Modify the values in one or more of the following fields:

ID: If you have need to reference this card outside of the user interface, specify an alphanumeric value here. If you do not assign a value, the Identity Server creates one for its internal use. The internal value is not persistent. Whenever the Identity Server is rebooted, it can change. A specified value is persistent.

Text: Specify the text that is displayed on the card to the user. This value, in combination with the image, should identify to the users, which provider they are logging into.

Image: Specify the image to be displayed on the card. Select the image from the drop-down list. To add an image to the list, click <*Select local image*>.

Show Card: Determine whether the card is shown to the user, which allows the user to select and use the card for authentication. If this option is not selected, the card is only used when a service provider makes a request for the card.

Passive Authentication Only: Select this option if you do not want the Identity Server to prompt the user for credentials. If the Identity Server can fulfill the authentication request without any user interaction, the authentication succeeds. Otherwise, it fails.

3 Click *OK* twice, then update the Identity Server.

7.10.2 Modifying the Authentication Card for SAML 1.1

When you create an identity provider, you must also configure an authentication card. After it is created, you can modify it.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 1.1* > *[Identity Provider]* > *Authentication Card.*

2 Modify the values in one or more of the following fields:

ID: If you have need to reference this card outside of the user interface, specify an alphanumeric value here. If you do not assign a value, the Identity Server creates one for its internal use. The internal value is not persistent. Whenever the Identity Server is rebooted, it can change. A specified value is persistent.

Text: Specify the text that is displayed on the card to the user. This value, in combination with the image, should identify to the users, which provider they are logging into.

Login URL: Specify an Intersite Transfer Service URL.The URL has the following format, where idp.sitea.novell.com is the DNS name of the identity provider, idp.siteb.novell.com is the name of the service provider, and idp.siteb.novell.com:8443/nidp/app specifies the URL that you want to users to access after a successful login:

```
https://idp.sitea.novell.com:8443/nidp/saml/idpsend?PID=https://
idp.siteb.novell.com:8443/nidp/saml/metadata&TARGET=https://
idp.siteb.novell.com:8443/nidp/app
```

For more information, see "Specifying the Intersite Transfer Service URL for the Login URL Option" on page 223.

If your identity provider is a Novell Identity Server and you know the ID specified for the target, you can use the following simplified format for the Login URL:

<URL for site a>?id=*<ID of target>*

For example:

```
https://idp.sitea.novell.com:8443/nidp/saml/idpsend?id=206test
```

The target and the target ID are specified in the service provider configuration at the identity provider. See "Configuring an Intersite Transfer Service Target for a Service Provider" on page 225.

Image: Specify the image to be displayed on the card. Select the image from the drop-down list. To add an image to the list, click <*Select local image*>.

Show Card: Determine whether the card is shown to the user, which allows the user to select and use the card for authentication. If this option is not selected, the card is only used when a service provider makes a request for the card.

3 Click *OK* twice, then update the Identity Server.

7.11 Using the Intersite Transfer Service

7.11.1 Understanding the Intersite Transfer Service URL

The Intersite Transfer Service is used by an identity provider to cause authentication to occur at a service provider that it trusts. The URLs for accessing the Intersite Transfer Service are different for each supported protocol (Liberty, SAML 1.1, and SAML 2.0). The Novell Access Manager identity and service provider components use the following format of the Intersite Transfer Service URL:

<identity consumer URL>?PID=*<entityID>*&TARGET=*<final_destination_URL>*

The *<identity_consumer_URL>* is the location of where the authentication request can be processed. For an Access Manager Identity Server, the URL is the Base URL of the server that is providing authentication, followed by the path to the protocol application being used for federation. For example:

SAML 1.1: `https://idp.sitea.novell.com:8443/nidp/saml/idpsend`

SAML 2.0: `https://idp.sitea.novell.com:8443/nidp/saml2/idpsend`

Liberty: `https://idp.sitea.novell.com:8443/nidp/idff/idpsend`

If a third-party server is providing the authentication, search its documentation for the format of this URL.

The *<entityID>* is the URL to the location of the metadata of the service provider. The scheme (http or https) in the *<entityID>* must match what is configured for the *<identity_consumer_URL>*. For SAML 1.1 and SAML 2.0, search the metadata for its entityID value. For Liberty, search the metadata for its providerID value. Novell Identity Servers acting as service providers have the following types of values:

SAML 1.1: `https://idp.siteb.novell.com:8443/nidp/saml/metadata`

SAML 2.0: `https://idp.siteb.novell.com:8443/nidp/saml2/metadata`

Liberty: `https://idp.siteb.novell.com:8443/nidp/idff/metadata`

If you are setting up federations with a third-party service provider, search its documentation for the URL or location of its metadata.

The *<final_destination_URL>* is the URL to which the browser is redirected following a successful authentication at the identity provider. If this target URL contains parameters (for example, `TARGET=https://login.provo.novell.com:8443/nidp/app?function_id=22166&Resp_Id=55321 &Resp_App_Id=810&security_id=0`), it must be URL encoded to prevent the URL from being truncated.

Examples with all three parts:

- **SAML 1.1:** `https://idp.sitea.novell.com:8443/nidp/saml/idpsend?PID=https://idp.siteb.novell.com:8443/nidp/saml/metadata&TARGET=https://eng.provo.novell.com/saml1/myapp`
- **SAML 2.0:** `https://idp.sitea.novell.com:8443/nidp/saml2/idpsend?PID=https://idp.siteb.novell.com:8443/nidp/saml2/metadata&TARGET=https://eng.provo.novell.com/saml2/myapp`
- **Liberty:** `https://idp.sitea.novell.com:8443/nidp/idff/idpsend?PID=https://idp.siteb.novell.com:8443/nidp/idff/metadata&TARGET=https://eng.provo.novell.com/liberty/myapp`

If you are configuring an Intersite Transfer Service URL for an Identity Server that is the identity provider and the service provider is either another Identity Server or a third-party server, you can simplify the Intersite Transfer Service URL to the following format:

<identity consumer URL>?id=*<user_definedID>*

For configuration information, see “Configuring an Intersite Transfer Service Target for a Service Provider” on page 225.

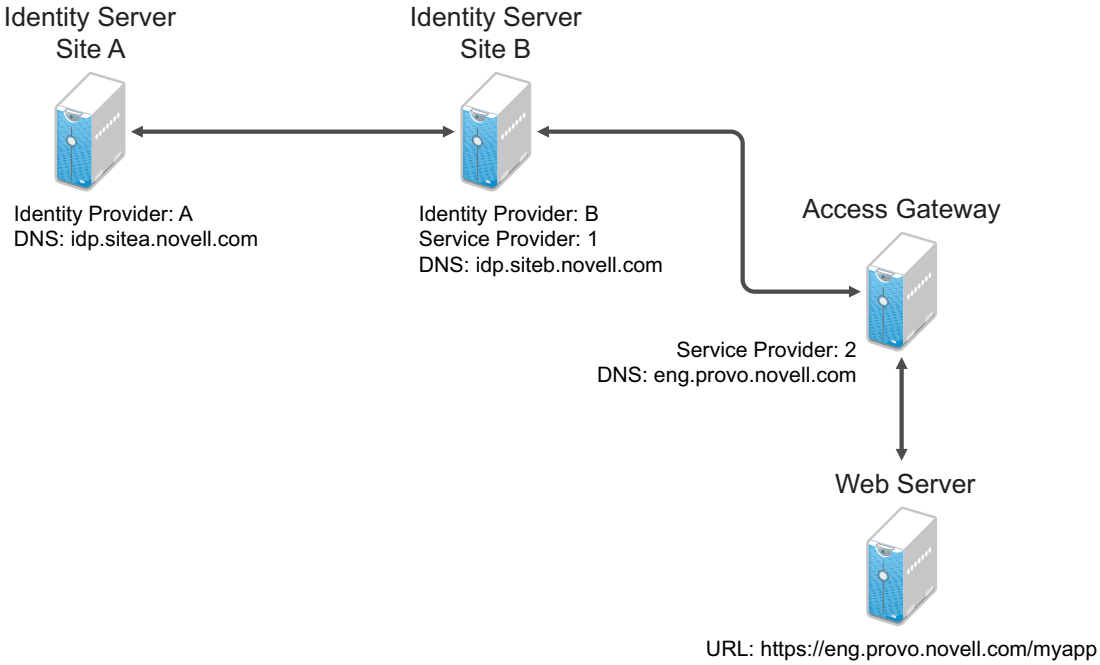
7.11.2 Specifying the Intersite Transfer Service URL for the Login URL Option

Liberty and SAML 2.0 support a single sign-on URL. Because SAML 1.1 does not support a single sign-on URL, you need to specify the Intersite Transfer Service URL in the *Login URL* option on the authentication card for the SAML 1.1 identity provider:

Figure 7-3 *SAML 1.1 Authentication Card*

In order for a card to appear as a login option, you must specify a *Login URL* and select the *Show Card* option. Figure 7-4 illustrates a possible configuration that requires the Intersite Transfer Service for the SAML 1.1 protocol.

Figure 7-4 *Federated Identity Configuration*



If you want a card to appear that allows the user to log in to Site A (as shown in Figure 7-3), you need to specify a value for the *Login URL* option.

Using the DNS names from Figure 7-4, the complete value for the *Login URL* option is as follows:

```
https://idp.sitea.novell.com:8443/nidp/saml/idpsend?PID=https://
idp.siteb.novell.com:8443/nidp/saml/metadata&TARGET=https://
idp.siteb.novell.com:8443/nidp/app
```

The following happens when this link is invoked:

1. The browser performs a Get to the identity provider (Site A).
2. If the identity provider (Site A) trusts the service provider (Site B), the identity provider prompts the user for authentication information and builds an assertion.
3. The identity provider (Site A) sends the user to the service provider (Site B), using the POST or Artifact method.
4. The service provider (Site B) consumes the assertion and sends the user to the TARGET URL (the user portal on Site B).

To configure the settings for the intersite transfer service, see Section 7.10.2, “Modifying the Authentication Card for SAML 1.1,” on page 220.

7.11.3 Using Intersite Transfer Service Links on Web Pages

The Intersite Transfer Service URL can be used on a Web page that provides links to various protected resources requiring authentication with a specific identity provider and a specific protocol. Links on this Web page are configured with the URL of the Intersite Transfer Service of the identity provider to be used for authentication. Clicking these links directs the user to the appropriate identity provider for authentication. Following successful authentication, the identity provider sends a SAML assertion to the service provider. The service provider uses the SAML assertion to verify authentication, and then redirects the user to the destination URL as specified in the TARGET portion of the Intersite Transfer Service URL.

Below are sample links that might be included on a Web page. These links demonstrate the use of SAML 1.1, SAML 2.0, and Liberty formats for the Intersite Transfer Service URL:

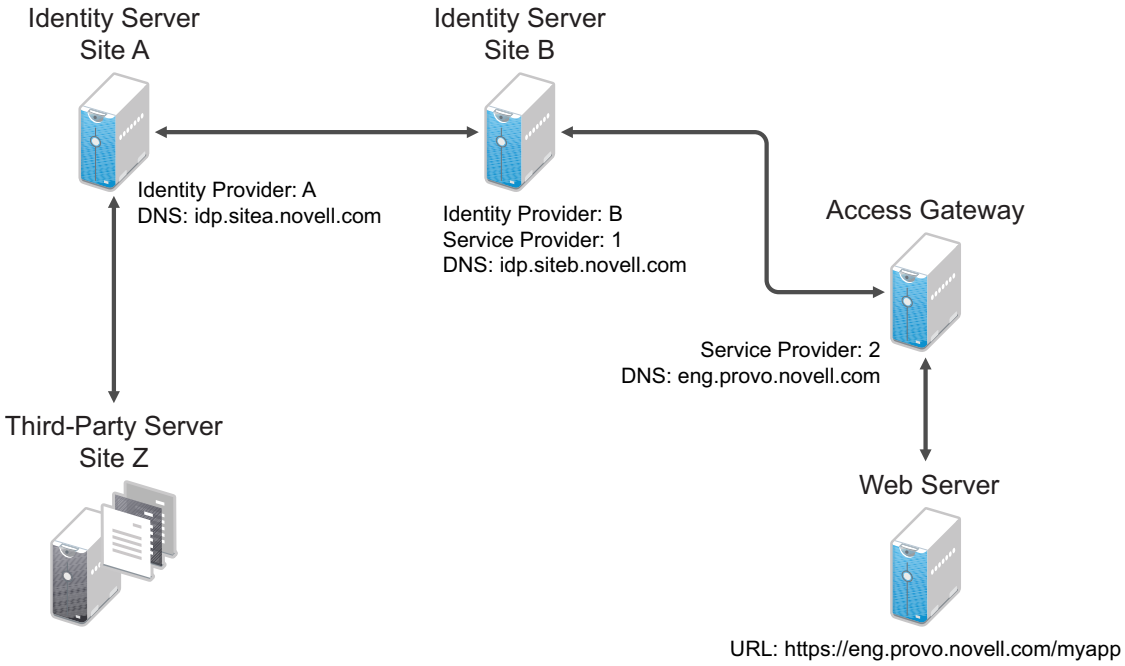
SAML 1.1: `<a href="https://idp.sitea.novell.com:8443/nidp/saml/idpsend?PID=https://idp.siteb.novell.com:8443/nidp/saml/metadata&TARGET=https://eng.provo.novell.com/saml1/myapp">SAML1 example</a>`

SAML 2.0: `<a href="https://idp.sitea.novell.com:8443/nidp/saml2/idpsend?PID=https://idp.siteb.novell.com:8443/nidp/saml2/metadata&TARGET=https://eng.provo.novell.com/saml2/myapp">SAML2 example</a>`

Liberty: `<a href="https://idp.sitea.cit.novell.com:8443/nidp/idff/idpsend?PID=https://idp.siteb.novell.com:8443/nidp/idff/metadata&TARGET=https://eng.provo.novell.com/liberty/myapp">Liberty example</a>`

Figure 7-5 illustrates a network configuration that could use these sample links.

Figure 7-5 *Using the Intersite Transfer Service URL*



In this example, Site Z places links on its Web page, using the Intersite Transfer Service URL of Site A. These links trigger authentication at Site A. If authentication is successful, Site A sends an assertion to Site B. Site B verifies the authentication and redirects the user to the myapp application that it is protecting.

7.11.4 Configuring an Intersite Transfer Service Target for a Service Provider

If you have created Web pages that have links that specify a Intersite Transfer Service URL (see "Using Intersite Transfer Service Links on Web Pages" on page 224), you can have the Identity Server control the TARGET parameter.

1. Click *Devices* > *Identity Servers* > *Edit* > *[Liberty, SAML1.1*, or *SAML 2.0]* > *[Service Provider]* > *Intersite Transfer Service*.
2. Fill in the following:

   ID: (Optional) Specify an alphanumeric value that identifies the target.

   If you specified an ID for the target, you can use this value to simplify the Intersite Transfer URL that must be configured at the service provider. This is the *<user_definedID>* value in the following format for the Intersite Transfer URL.

   ```
   <identity consumer URL>?id=<user_definedID>
   ```

   The ID specified here allows the Identity Server to find the service provider's metadata. The *Target* option on this page allows you to omit the TARGET parameter from the Intersite Transfer URL.

   Target: Specify the URL of the page that you want to display to users when they authenticate with an Intersite Transfer URL.The behavior of this option is influenced by the *Allow any target* option.

If you are using the target ID as part of the Intersite Transfer URL and did not specify a target in the URL, you need to specify the target in this field.

Allow any target: If this option is selected, the user can use the target that was specified in the Intersite Transfer URL. If this option is not selected, the target value in the Intersite Transfer URL is ignored and the user is sent to the URL specified in the *Target* option.

3 Click *OK* twice.

4 Update the Identity Server.

Accessing The Intersite Tansfer URL With A Specific Contract

Novell Identity Provider server talks to SAML Service Provider through intersite transfer URL ID, which is defined in intersite transfer URL configuration.

For Example: `https://idp.novell.com/nidp/saml2/idpsend?id=samlsp`

If we are using default Identity Provider contract then it works fine. If the administrator wants to define a contract to execute on Identity Provider when talking to the specific Service Provider, then they use the following URL:

`https://idp.novell.com/nidp/app?id=contract&target=https://idp.novell.com/nidp/saml2/idpsend?id=samlsp`.

The above URL works fine if the user is prompted with the login page tied to contract.

If the user is already logged in and the URL is bookmarked and accesses the URL, then the Idenity Provider portal page is displayed instead of the Service Provider target page. The users should be provided with the option to define a contract on the intersite transfer URL, where users who are already authenticated can go to the Service Provider after accessing the link rather than the Identity Provider portal.

For Example:

`https://idp.novell.com/nidp/app/login?id=contract&target=https://idp.novell.com/nidp/saml2/idpsend?id=samlsp`

is changed to

`https://idp.novell.com/nidp/app/login`

Configuring CardSpace

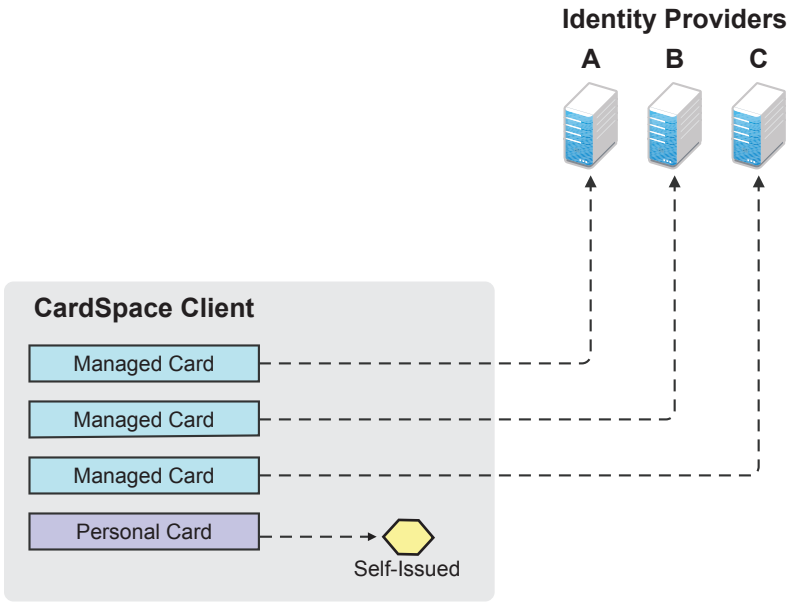
8

This section describes the following Microsoft CardSpace configuration tasks:

8.1 Overview of the CardSpace Authentication Process

CardSpace puts the users in control of managing the cards that they want to use for identity information and credentials. With a CardSpace client, the users can create managed cards and personal cards for authentication to the Novell Identity Server. Figure 8-1 illustrates this process.

Figure 8-1 *The Relationship between Cards and Providers*



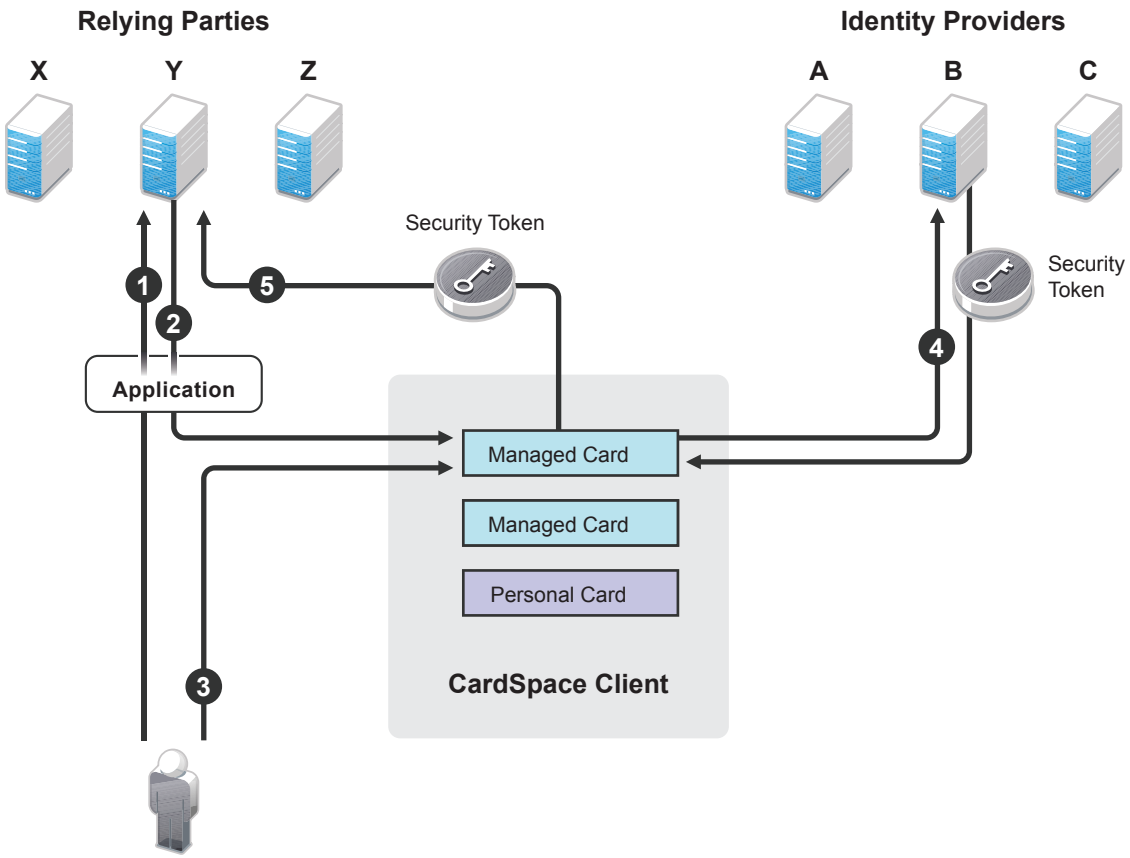
Managed cards come from an identity provider. When the users interact with the Identity Server, they can install a managed card from the Identity Server into the CardSpace client. The managed card provides metadata to CardSpace about how to interact with the Identity Server, which includes the available attributes (claims).

Personal cards are created with the CardSpace client software, and the user decides which attributes are available.

The purpose of a card is to define the source for the identity, the provider of the authentication token, and the credentials provided in the token. Figure 8-1 illustrates that the provider for the identity and token can be either an identity provider when a managed card is selected or the CardSpace client when a personal card is selected.

Figure 8-2 illustrates the process when a relying party requests a token.

Figure 8-2 *Using a Card for Authentication*



1. The user requests access to an application, and the application sends the request to the relying party.
2. The relying party returns the security token requirements, which include the issuer ID, the required attributes, and the token type to CardSpace.
3. The CardSpace client software highlights the cards that meet the requirements, and the user selects the card to use.

4. The CardSpace client software requests a security token from its configured trusted identity provider, and the identity provider returns the security token.
5. The CardSpace client software presents the token to the relying party, and if it matches the requirements, the user is granted access.

The Novell Identity Server can be configured to act as relying party or as an identity provider. It can be configured to accept the following types of cards for authentication: personal cards, managed cards, and managed cards backed by personal cards.

8.2 Prerequisites for CardSpace

- ❑ Your Identity Server cluster configuration must be configured for HTTPS. For configuration information, see "Enabling SSL Communication" in the *Novell Access Manager 3.1 SP2 Setup Guide*.
- ❑ CardSpace requires high encryption. However, export laws prevent Access Manager from shipping with the high encryption library for JRE. To add this library, see Section 8.2.1, "Enabling High Encryption," on page 229.
- ❑ Clients need to be configured with a CardSpace client. See Section 8.2.2, "Configuring the Client Machines for CardSpace," on page 230.
- ❑ Enable the Liberty Personal Profile. The default attribute set created for CardSpace is dependent upon this profile.

  Click *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*. Select the *Personal Profile*, then click *Enable* > *Apply*. Update the Identity Server.
- ❑ (Recommended) Enable Identity Server logging while you are setting up CardSpace. Set the Component File Logger Levels of STS and CardSpace to debug. For more information, see Section 14.3, "Configuring Component Logging," on page 324.
- ❑ (Optional) If you are configuring an Identity Server to be an identity provider with managed cards, you need a second Identity Server configured to be a relying party.

8.2.1 Enabling High Encryption

To enable high encryption, you need to replace the `US_export_policy.jar` and `local_policy.jar` files. The Identity Server that is going to be the relying party and the Identity Server that is going to be the identity provider need to be enabled for high encryption.

1. Download the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6 (jce_policy-6.zip) (http://java.sun.com/javase/downloads/index.jsp).
2. Extract the files.
3. Copy the `US_export_policy.jar` and `local_policy.jar` files to the security directory for the JRE. They should replace the existing files:
   - **Linux Identity Server:** `/opt/novell/java/jre/lib/security`
   - **Windows Server 2003 Identity Server:** `\Program Files\Novell\jre\lib\security`
   - **Windows Server 2008 Identity Server:** `\Program Files (x86)\Novell\jre\lib\security`

4 Restart Tomcat.

- **Linux Identity Server:** Enter the following command:

  ```
  /etc/init.d/novell-tomcat5 restart
  ```

- **Windows Identity Server:** Enter the following commands:

  ```
  net stop Tomcat5
  net start Tomcat5
  ```

8.2.2 Configuring the Client Machines for CardSpace

The client machines require a CardSpace card selector application. They also need to be configured to trust the machine that is acting as an identity provider.

- "Configuring Windows Clients for CardSpace" on page 230
- "Configuring Linux Clients for CardSpace" on page 231

Configuring Windows Clients for CardSpace

Windows clients require the Microsoft .NET Framework 3.5 service pack, and Internet Explorer needs to be configured to trust the identity providers that supply managed cards.

1 (Conditional) Install the Microsoft .NET Framework 3.5 service pack.

For Windows 7 and Vista clients, this is included with the operating system.

For XP clients, you need to download and install it:

1a Download the package. See Microsoft .NET Framework 3.5 (http://www.microsoft.com/downloads/details.aspx?FamilyId=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en)

1b Install the package.

1c To verify that it has been installed, click *Control Panel* > *Add and Remove Programs*, then search for a Microsoft .NET Framework 3.5 entry.

2 (Conditional) If you are using Access Manager generated certificates, you need to install the trusted root certificate of the Identity Server CA so that Internet Explorer trusts the Identity Server.

You must be an administrator user to complete these steps.

2a In Internet Explorer, enter the base URL of the Identity Server.

2b Click *Continue to this website*.

2c In the URL line, click *Certificate Error* > *View Certificates*.

The Certificate Information page displays information about the Identity Server server certificate.

2d Click *Certification Path*, select the root CA certificate, then click *View Certificate*.

The Certificate Information page displays information about the root CA certificate.

2e Click *Install Certificate* > *Next*.

2f Select *Place all certificates in the following store*, then click *Browse*.

2g Select to *Show physical stores*, scroll to the *Trusted Root Certification Authorities*, open it, select *Local Computer*, then click *OK*.

2h Click *Next* > *Finish* > *OK*.

2i Close the browser.

2j To verify that the correct certificate was installed, open the browser, then enter the base URL of the Identity Server.

The certificate error should not appear in the URL line.

Configuring Linux Clients for CardSpace

The following instructions are for Linux clients running SUSE Linux Enterprise Server (SLES) 10. They explain how to use the Bandit DigitalMe card selector, including how to download it, install it, and configure it so that it trusts the Identity Server.

1. Verify that you have updated Firefox to 2.x or later. DigitalMe does not work with Firefox 1.5.x.
2. In Firefox, access the Bandit Card site by entering the following URL:

   `http://cards.bandit-project.org`
3. Click *Download a selector*, then select to download the selector for OpenSUSE.
4. Scroll to the bottom of the page, and install the Firefox add-on.

   4a Click *Download DigitalMe add-on for Firefox (All Platforms)*.

   4b If you haven't enabled the Bandit site to install plug-ins, click *Edit Options*, then enable the site and install the add-on.
5. Download the appropriate selector for your OS. For SLES 10 with 32-bit hardware, select *Download DigitalMe for SUSE Linux Enterprise 10 (i586)* and save it as a file.
6. Close Firefox.
7. Open the download and install it.
8. Export the public key certificates of the Identity Server. You need both the CA and server certificates.

   The following instructions explain how to log in to the Administration Console from the client machine with DigitalMe and export the certificates to the required directory.

   8a From a browser on the DigitalMe machine, log into the Administration Console.

   8b Click *Security* > *Certificates*.

   8c Click the name of the Identity Server certificate, then click *Export Public Certificate* > *DER File*.

   8d Select to save the file to disk, then click *OK*.

   8e Click *Close*, then click *Trusted Roots*.

   8f Click the name of the trusted root (the default name is *configCA*), then select to *Export Public Certificate* > *DER File*.

   8g Select to save the file to disk, then click *OK*.

   8h Copy the two certificate files to the following directory:

   `/usr/share/digitalme/certs`
9. From the Application Browser, start the DigitalMe card selector.
10. At the prompt to create a default keying, enter a password, reenter the password, then click *OK*.

8.3 CardSpace Configuration Scenarios

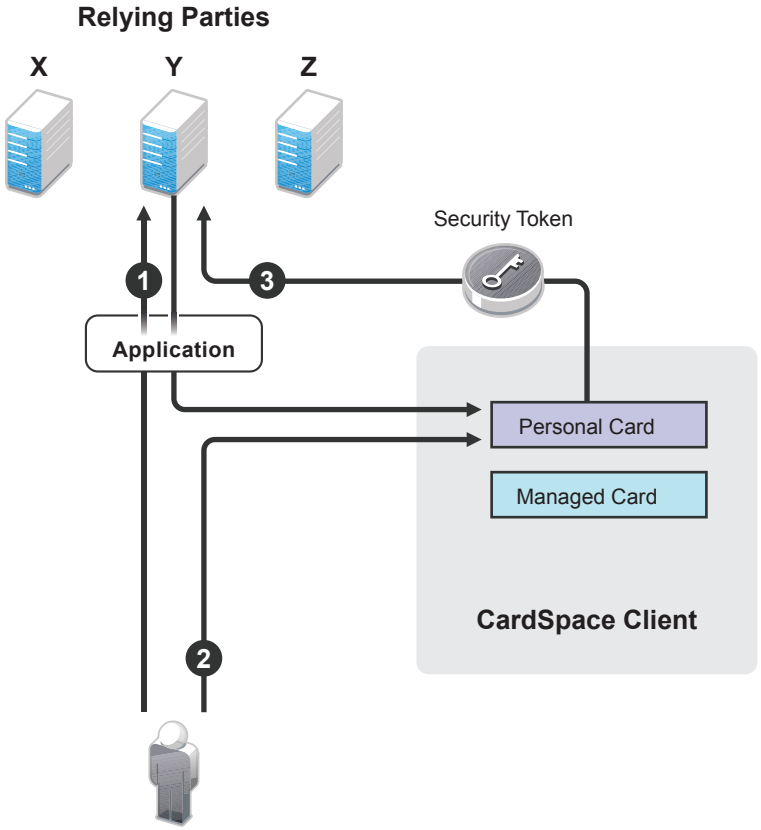
The following sections explain the configuration process for three common ways of using CardSpace for authentication.

- Section 8.3.1, “Authenticating with a Personal Card,” on page 232
- Section 8.3.2, “Authenticating with a Managed Card,” on page 234
- Section 8.3.3, “Authenticating with a Managed Card Backed by a Personal Card,” on page 238

8.3.1 Authenticating with a Personal Card

The following scenario explains how to configure the Identity Server to be a relying party and then allow the user to log in to the Identity Server by using a personal card. Figure 8-3 illustrates this process:

Figure 8-3 *Using a Personal Card to Authenticate to a Relying Party*



1. The user requests authentication at the Identity Server by entering the base URL of the Identity Server in the browser. This opens the user portal application.
2. The user selects an authentication card that requires a personal card.
3. From the available cards in CardSpace, the user selects the card that meets the security requirements, and the CardSpace client software sends it to the Identity Server.

To configure this scenario:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit*.
2. In the *Enabled Protocols* section, enable *STS* and *CardSpace*.
3. Click *CardSpace* > *Authentication Card*, then fill in the following fields:

   ID: (Optional) Leave this field blank.

   Text: Specify the text that is displayed on the card to the user, for example, `CardSpace`.

   Image: Select the image from the drop-down list. For CardSpace, you can use the default CardSpace image or any other image in the list.

   Show Card: Enable the *Show Card* option. The Identity Server then displays this card as a login option.
4. In the Profiles section, click *New*, then fill in the following fields:

   Name: Specify a display name for the profile, such as `Personal Card`.

   ID: (Optional) Leave this field blank.

   Text: Specify the text that is displayed on the card to the user for this profile, such as `Personal Card`.

   Issuer: From the drop-down list, select *Personal Card.*

   Token Type: SAML 1.1 is displayed as the token type for the assertion.
5. Click *Next*, then specify the attributes for the personal card.

   Attribute set: Select the *CardSpace* attribute set.

   Required attributes: From the *Available attribute* list, select the attributes that you want the card to return and move them to the *Required attribute* list.

   For this scenario, move *Common First Name* and *Personal Private Identifier* to the *Required attribute* list. The *Personal Private Identifier* attribute should always be in the required list.

   Optional attributes: From the *Available attribute* list, select the attributes that the card can return, but is not required to return, and move them to the *Optional attribute* list.

   For this scenario, move *Common Last Name*.
6. Click *Next*, then specify the user identification method.

   Satisfied contracts: (Optional) For this scenario, do not select a contract.

   Allow federation: Enable this option so that the personal card can be linked with the user's account. If you do not enable this option, the user is always prompted for credentials.

   Authenticate: Select *Authenticate* for the user identification method. This prompts the user for a name and a password the first time the card is used for authentication.
7. Click *Finish* > *OK*.
8. Update the Identity Server.
9. In the browser, enter the base URL of the Identity Server.
10. Select the authentication card you have created.

    The CardSpace selector opens.
11. Create a personal card that meets the requirements of the authentication profile. Provide a value for First Name claim and optionally for the Last Name.
12. Save the card, then click *Send*.

13 Enter the username and a password for an account in the user store.

You are logged in. On subsequent logins, you do not need to enter the username and password.

A personal card can be used to access resources protected by an Access Gateway, but it must be used with a managed card. For this scenario, you need to complete the tasks in the following sections:

- Section 8.3.2, "Authenticating with a Managed Card," on page 234
- Section 8.3.3, "Authenticating with a Managed Card Backed by a Personal Card," on page 238
- Section 8.6, "Using CardSpace Cards for Authentication to Access Gateway Protected Resources," on page 246

For more information about configuring the Identity Server to be a relying party and the other available options, see Section 8.4, "Configuring the Identity Server as a Relying Party," on page 239.

8.3.2 Authenticating with a Managed Card

To use a managed card, you need both a relying party and an identity provider as illustrated in Figure 8-2 on page 228. If you completed the steps in Section 8.3.1, "Authenticating with a Personal Card," on page 232, you have set up an Identity Server as a relying party. The following scenario explains how to set up a second Identity Server to be the identity provider. It also explains how to configure a trusted relationship between the relying party and the identity provider, so that a user can authenticate to the relying party with a managed card.

- "Prerequisite" on page 234
- "Configuring a CardSpace Identity Provider" on page 235
- "Creating and Installing a Managed Card" on page 235
- "Configuring the Relying Party to Trust an Identity Provider" on page 236
- "Logging In with the Managed Card" on page 238

These sections describe only a few of options available for configuring the Identity Server as a CardSpace identity provider. For information about all the available options, see Section 8.5, "Configuring the Identity Server as an Identity Provider," on page 243.

Prerequisite

For CardSpace and managed cards, you need to make sure that the SSL certificate and the signing certificate of the Identity Server use the same name for the certificate's subject name. When you configured the Identity Server for SSL, you replaced the default SSL certificate with a certificate that uses the DNS name of the Identity Server as the subject name. For CardSpace, you need to replace the default signing certificate. You can use the same certificate for signing as you did for SSL.

Both Identity Server that is the relying party and the Identity Server that is the identity provider need a signing certificate that uses the DNS name of the Identity Server as the subject name.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Security*.

2 In the *Keys and Certificate* section, click *Signing*.

3 Click *Replace*.

4 In the Replace pop-up, click the *Select Certificate* icon, select the certificate you created for SSL, then click *OK*.

5 When the certificate appears in the Certificate box, click *OK*, then click *Close*.

6 Update the Identity Server.

Configuring a CardSpace Identity Provider

When you configure an Identity Server to be a CardSpace identity provider, you need to create a managed card template. Users can then use the template to create and install a managed card in their card selector.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace*.

2 Click *Managed Card Templates* > *New*, then fill in the following fields:

Name: Specify a display name for the template.

Description: Specify the text to be displayed on the card. This can contain information about how the card can be used or the type of resource that can be accessed with the card.

Image: Specify the image to be displayed on the card. Select the image from the drop-down list. To add an image to the list, click *Select local image*. The default image is the Novell Card.

Require Identification of Relying Party in Security Token: Select this option to require the relying party to provide identification when it requests a security token. For this scenario, do not enable this option because the instructions haven't explained how to configure this option for the relying party.

Allow Users to Back a Managed Card Using a Personal Card: Select this option to allow users to back a managed card with a personal card. If this option is not selected, you cannot complete the steps in Section 8.3.3, "Authenticating with a Managed Card Backed by a Personal Card," on page 238.

3 Click *Next*, then fill in the following fields:

Attribute set: From the list of available sets, select the CardSpace attribute set.

Selected claims: From the list of available claims, select the attributes for the managed card and move them to the list of selected claims.

Do not remove the *Personal Private Identifier* claim. Add the *Common First Name* claim.

4 Click *Finish*.

5 Click *STS* > *Authentication Methods*.

6 Move the *Secure Name/Password - Form* method to the *Methods* list.

7 Click *OK*.

8 Update the Identity Server.

9 Continue with "Creating and Installing a Managed Card" on page 235.

Creating and Installing a Managed Card

The following instructions assume you are on a Windows client. The procedure is very similar to what is required on a Linux client and should be easily adapted.

1 In Internet Explorer on the client machine, enter the base URL of the Identity Server that is acting as the identity provider.

2 Select the Secure Name/Password card, then log in to the Identity Server.

3 Click *New Card*, then click the *Managed Card Templat*e.

The card displays the required claims.

4 Specify a name for the card, then click *Create Card*.

5 Click *Open.*

CardSpace opens.

6 Click *Install and Exit*.

The managed card is installed.

7 Log out and close the browser.

8 Continue with "Configuring the Relying Party to Trust an Identity Provider" on page 236.

Configuring the Relying Party to Trust an Identity Provider

A trusted provider is an issuer of authentication tokens that you want to strongly trust. The provider has given you its issuer ID and its public key for the signing certificate. Tokens issued from this trusted provider are validated by using the public key certificate.

To configure a trusted relationship between the relying party and the identity provider, you need to create a trusted provider configuration for the identity provider. You also need to either modify an existing authentication profile or create a profile that includes the trusted provider as an issuer of security tokens.

To create a trusted provider configuration for the Identity Server acting as the identity provider, you need to know the base URL of the Identity Server and have a file containing the public key of the signing certificate of the Identity Server.

1 To obtain the public key certificate of the identity provider:

1a Log in to the Administration Console of the identity provider.

1b Click *Security* > *Certificates*.

1c Click the certificate you have created for the Identity Server to use for SSL and signing.

1d On the certificate page, click *Export Public Certificate* > *DER File*, then save the certificate to a file.

1e Copy this file to a location available to the Administration Console for the relying party.

2 To create a trusted provider configuration for the identity provider:

2a Log in to the Administration Console for the relying party.

2b Click *Devices* > *Identity Servers* > *Edit* > *CardSpace*.

2c Click *Trusted Providers* > *New*, then fill in the following fields:

Name: Specify a display name for the identity provider. This name appears in the list of trusted providers that you can select for an authentication card profile. You might want to use part of the DNS name of the identity provider.

Source: This line specifies that the Provider ID is entered manually.

Provider ID: Specify the issuer ID of the trusted provider. For an Identity Server cluster configuration, the issuer ID is the base URL of the Identity Server plus the following path:

`/sts/services/Trust`

For example, if the base URL is `https://test.lab.novell.com:8443/nidp`, the Provider ID is the following value:

`https://test.lab.novell.com:8443/nidp/sts/services/Trust`

Identity Provider: Click *Browse* to browse for and select the certificate that you exported for the identity provider.

2d Click *Next* > *Finish*.

3 To create a profile that allows this trusted provider to be an issuer of security tokens, click *Authentication Card*.

The following steps explain how to create a new profile for the trusted provider. This allows you to see how a CardSpace authentication card can be configured for multiple profiles.

3a Click *New*, then fill in the following fields:

Name: Specify a display name for the profile that indicates which trusted provider is going to use the profile.

ID: (Optional) Leave this field blank.

Text: Specify the text that is displayed on the card to the user for this profile. If the user knows about the identity provider, this should help the user identify the provider.

Issuer: From the drop-down list, select the name of the trusted provider.

Token Type: SAML 1.1 is displayed as the token type for the assertion.

3b Click *Next*, then specify the attributes for the personal card.

Attribute set: Select the *CardSpace* attribute set.

Required attributes: From the *Available attribute* list, select the attributes that you want the card to return and move them to the *Required attribute* list.

For this scenario, move *Common First Name* and *Personal Private Identifier* to the *Required attribute* list. The *Personal Private Identifier* attribute should always be in the required list.

Optional attributes: From the *Available attribute* list, select the attributes that the card can return, but is not required to return, and move them to the *Optional attribute* list. For this scenario, do not select any optional attributes.

3c Click *Next*, then specify the user identification method.

Satisfied contract: (Optional) For this scenario, do not select a contract.

Allow federation: Enable this option so that the managed card can be linked with the user's account. If you do not enable this option, the user is always prompted for credentials.

Authenticate: Select *Authenticate* for the user identification method. This prompts the user for a name and a password the first time the card is used for authentication.

4 To add a trusted root to a trust store, click *Security* > *Certificates*.

The Certificates page is displayed.

4a Click *Trusted Roots* > *Auto-Import From Server*.

In the pop-up dialog box, fill in the following fields:

Server IP/DNS: Specify the server IP address or DNS name for the identity provider.

Server Port: Specify 8443 for the server port number.

Certificate name: Specify a name for the certificate.

4b Click *OK*.

4c Select the imported certificate, then click *Add Trusted Roots to Trust Stores*.

4d In the *Trust store(s)* field, click the *Select Keystore* icon.

4e Select *NIDP-truststore*, then click *OK* > *OK*.

5 Update the Identity Server.

6 Continue with "Logging In with the Managed Card" on page 238.

Logging In with the Managed Card

1 In the browser on the client machine, enter the base URL of the Identity Server acting as the relying party.

2 On the CardSpace card, click the *Card Options* icon in the top right corner.

3 Select the profile option for the managed card.

4 When the CardSpace application opens, select the managed card you imported, then click *Send*.

5 In the CardSpace application, enter the password for the user, then click *OK*.

6 When prompted by the Identity Server, enter the name and password.

On subsequent logins, CardSpace prompts you for a password, but the Identity Sever uses the card for authentication. For single sign-on with the managed card, you need to back it with a personal card. Continue with Section 8.3.3, "Authenticating with a Managed Card Backed by a Personal Card," on page 238.

Managed cards can be used to access resources protected by the Access Gateway. For configuration information, see Section 8.6, "Using CardSpace Cards for Authentication to Access Gateway Protected Resources," on page 246.

8.3.3 Authenticating with a Managed Card Backed by a Personal Card

The following configuration assumes that you have completed the configuration steps for Section 8.3.2, "Authenticating with a Managed Card," on page 234 and that you enabled the *Allow Users to Back a Managed Card Using a Personal Card* option. This configuration scenario uses the managed card that you have created and explains how to install a new instance of it and back it with a personal card.

1 In a browser on the client machine, enter the base URL of the Identity Server acting as the identity provider.

2 Select the *Secure Name/Password* card, then log in to the Identity Server.

3 Click *New Card*, then click the *Managed Card Templat*e.

4 Specify a name for the card, then enable the *Use Personal Card For Authentication* option.

5 When CardSpace opens, select a personal card, then click *Send*.

6 On the New Card page, click *Create Card*.

7 Click *Open*.

CardSpace opens.

8 Click *Install and Exit*.

The managed card backed by a personal card is installed.

9 Log out and close the browser.

10 In the browser, enter the base URL of the Identity Server acting as the relying party.

11 Select the CardSpace card.

12 In your card selector, select the managed card that is backed by a personal card, then click *Send*.

13 When prompted, enter the username and password, and log in.

On subsequent logins, you can use the card to log in without entering any credentials.

14 Click the *Federation* tab.

It displays the name of the card that you used to log in with and allows you to break the federation with the personal card. When you break the federation, you must supply credentials to log n.

For information on using this card with resources protected by the Access Gateway, see Section 8.6, "Using CardSpace Cards for Authentication to Access Gateway Protected Resources," on page 246

8.4 Configuring the Identity Server as a Relying Party

When the Identity Server is acting as the relying party, you need to define how you want the user to authenticate. This involves defining who can issue the credentials and what credentials are required.

For a basic setup, see "Configuring the Relying Party to Trust an Identity Provider" on page 236.

8.4.1 Defining an Authentication Card and Profile

The authentication card defines the visual aspects of the card. An authentication card profile defines the parameters for accessing CardSpace. Multiple profiles can be created for the authentication card, and the user can select which profile to use for authentication.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace*.

2 Click *Authentication Card*, then fill in the following fields:

ID: (Optional) Specify an alphanumeric value that identifies the card. If you need to reference this card outside of the user interface, you need to specify a value here. If you do not assign a value, the Identity Server creates one for its internal use.

Text: Specify the text that is displayed as the card name to the user, such as CardSpace.

Image: Select the image from the drop-down list. For CardSpace, you can use the default CardSpace image or any other image in the list. To add a new image, click *Select local image*. For more information on how to add an image, see Section 6.5, “Adding Authentication Card Images,” on page 184.

Show Card: Select this option when you want the Identity Server to display the card as a login option. Deselect this option when you want to prevent users from using this card and any of its authentication profiles.

3 In the *Profiles* section, click *New*, then fill in the following fields:

Name: Specify a display name for the profile.

ID: (Optional) Specify an alphanumeric value that identifies the card. If you need to reference this card outside of the Administration Console, you need to specify a value here. If you do not assign a value, the Identity Server creates one for its internal use.

Text: Specify the text that references the profile when more than one profile has been defined.

Issuer: From the drop-down list, select one of the following:

- **Any Trusted or Untrusted Provider or Personal Card:** Specifies that the issuer of the card can be a managed card from any provider or can be a personal card. This option allows all cards in the card selector to be selected.
- **Personal Card:** Specifies that the issuer must be a personal card from a card selector.
- **Any Trusted Provider or Personal Card:** Specifies that the card can be either a personal card or a managed card from any trusted provider. A trusted provider is a provider that is listed in the trusted provider list. See Section 8.4.2, “Defining a Trusted Provider,” on page 241.

  This option allows all cards in the card selector to be selected. The Identity Server enforces the trusted provider requirement when the card is sent.
- **<Provider Name>:** Specifies that the card must be a managed card from the specified provider. To add a trusted provider, see Section 8.4.2, “Defining a Trusted Provider,” on page 241.

Token Type: SAML 1.1 is displayed as the token type for the assertion.

If you are using CardSpace to allow access to Access Gateway protected resources, you must ensure that the contract specified for a protected resource is satisfied by an authentication profile.

4 Click *Next*, then specify the attributes for the card profile.

Attribute set: Select the CardSpace attribute set.

Required attributes: From the *Available attribute* list, select the attributes that you want the card to return and move them to the *Required attribute* list.

Move *Common First Name* and *Personal Private Identifier* to the *Required attribute* list.

Optional attributes: From the *Available attribute* list, select the attributes that the card can return, but is not required to return, and move them to the *Optional attribute* list.

5 Click *Next*, then specify the user identification method.

Satisfied contracts: (Optional) Move the contract that you want this profile to satisfy from the list of available contracts to the *Satisfied contract* list.

Allow federation: Allows the CardSpace card to be linked with a user account. If you do not select this option, the user is always prompted for credentials.

User Identification Methods: If you enable federation, the user identification method determines how the card is linked to a user account and allows the association to be saved. If you do not enable federation, a user identification method allows the card to be linked with an account, but the association is not saved. Select one of the following methods:

- **Do nothing:** Select this option to allow the user to authenticate without creating an association with a user account. This option cannot be used when federation is enabled.
- **Authenticate:** Select this option when you want to use login credentials. This option prompts the user to log in to the service provider.
  - **Allow 'Provisioning':** Select this option to allow users to create an account when they have no account on the service provider.

    This option requires that you specify a user provisioning method, which defines the required attributes for setting up a user account. See Section 11.3, "Defining the User Provisioning Method," on page 286.
- **Provision Account:** Select this option when the users on the identity provider do not have accounts on the service provider. This option allows the service provider to trust any user that has authenticated to the trusted identity provider.

  This option requires that you specify a user provisioning method, which defines the required attributes for setting up a user account. See Section 11.3, "Defining the User Provisioning Method," on page 286.
- **Attribute matching:** Select this option when you want to use attributes to match an identity server account with a service provider account. This option requires that you specify a user matching method. See Section 11.1.2, "Configuring the Attribute Matching Method for Liberty or SAML 2.0," on page 283.
  - **Prompt for password on successful match:** Select this option to prompt the user for a password when the user's name is matched to an account, to ensure that the account matches.

6 (Conditional) If you have selected a method that requires account provisioning or attribute matching, click the icon for *Provisioning Settings* or *Attribute Matching Settings*. For instructions, see Section 11.3, "Defining the User Provisioning Method," on page 286 or Section 11.1.2, "Configuring the Attribute Matching Method for Liberty or SAML 2.0," on page 283.

7 Click *Finish* > *OK*.

8 Restart the Identity Server. Stopping and starting the Identity Server also updates its configuration:

8a On the Identity Servers page, select the server, then click *Stop* > *OK*.

8b When the health turns red, select the server, then click *Start*.

9 Continue with Section 8.4.2, "Defining a Trusted Provider," on page 241.

8.4.2 Defining a Trusted Provider

You need to create a trusted provider for each server you want to explicitly trust as an identity provider. If your users are going to use only personal cards for authentication or it explicit trust is not required, you do not need to create a trusted provider configuration.

The authentication profile allows you to select an option to trust any provider, including untrusted providers. For a secure system, you need to identify the providers you want to trust and create a configuration for them. To create a trusted provider, you need to obtain the issuer ID of the provider and the public key certificate for signing certificate from the provider's administrator.

For an Identity Server cluster, the issuer ID is the base URL of the Identity Server plus the following path:

```
/sts/services/Trust
```

For example, if the base URL is `https://test.lab.novell.com:8443/nidp`, the Provider ID is the following value:

```
https://test.lab.novell.com:8443/nidp/sts/services/Trust
```

This section explains the following:

- "Creating a Trusted Provider Configuration" on page 242
- "Managing the Trusted Provider Configuration" on page 242

Creating a Trusted Provider Configuration

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace*.
2. On the Trusted Providers page, click *New*, then fill in the following fields:

   Name: Specify a display name for the provider. This name appears in the list of trusted providers that you can select for an authentication card profile.

   Source: This line specifies that the Provider ID is entered manually.

   Provider ID: Specify the issuer ID of the trusted provider. For an Identity Server cluster when the base URL is `https://test.lab.novell.com:8443/nidp`, the Provider ID is the following value

   ```
   https://test.lab.novell.com:8443/nidp/sts/services/Trust
   ```

   For a third-party identity provider, you need to obtain the issuer ID from the provider.

   Signing Certificate: Import the certificate by clicking *Browse*. Find the signing certificate file, click *Open* to import it, then click *Next*.
3. To confirm the signing certificate, click *Finish*.

Managing the Trusted Provider Configuration

You can modify the name of the configuration, view and edit the metadata, view and reimport the signing certificate.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace*.
2. On the *Trusted Providers* page, click the name of a trusted provider.
3. To change the name of the trusted provider, specify a new name on the *Configuration* page, then click *Apply*.
4. To view or edit the metadata, click *Metadata*.

5 To modify the Provider ID or to import a new signing certificate, click *Edit*.

5a (Optional) To change the Provider ID, enter a new value or modify the current value.

5b (Optional) To import a new signing certificate, click *Browse*, find the certificate file, click *Open* to import it, then click *Apply*.

6 To view the signing certificate, click *Certificates*.

7 (Conditional) If you made any modifications, update the Identity Server.

8.4.3 Cleaning Up Identities

When acting as a relying party, you can set limits for how long an identity can remain unused before the identity is automatically defederated. The default value is 90 days. You can specify a value from 0 to 365 days. To configure this value:

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace*.

2 Click *Configuration*.

3 Specify a value for the relying party maximum age.

4 Click *Apply*, then update the Identity Server.

8.4.4 Defederating after User Portal Login

If you want to remove the federation link on a card so you are prompted for login credentials the next time you use it, you need to defederate the card.

1 Log in to the user portal.

2 In your authentication card section, select the card you used to authenticate.

3 Click the options icon.

4 To defederate this account, select the *defederate* option.

8.5 Configuring the Identity Server as an Identity Provider

When the Identity Server is acting as a CardSpace identity provider, you need to configure the Identity Server's certificates to support CardSpace, configure the underlying STS to support CardSpace, and create a managed card template:

- Section 8.5.1, "Replacing the Signing Certificate," on page 244
- Section 8.5.2, "Configuring STS," on page 244
- Section 8.5.3, "Creating a Managed Card Template," on page 245

For a basic set up, see Section 8.3.2, “Authenticating with a Managed Card,” on page 234.

8.5.1 Replacing the Signing Certificate

For CardSpace and managed cards, you need to make sure that the SSL certificate and the signing certificate of the Identity Server use the same name for the certificate’s subject name. When you configured the Identity Server for SSL, you replaced the default SSL certificate with a certificate that uses the DNS name of the Identity Server as the common name in the subject name of the certificate. For CardSpace, you need to replace the default signing certificate. You can use the same certificate for signing as you did for SSL or you can use different certificate, if the full subject name is the same as the certificate you have configured for SSL.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Security*.
2. In the *Keys and Certificate* section, click *Signing*.
3. Click *Replace*.
4. In the Replace pop-up, click the *Select Certificate* icon, select the certificate with the correct subject name, then click *OK*.
5. When the certificate appears in the *Certificate* box, click *OK*, then click *Close*.
6. Update the Identity Server.

8.5.2 Configuring STS

CardSpace relies on the Security Token Service (STS), which controls what claims are available, what authentication method can be used to validate the credentials on the card, and whether a name identifier is added to the SAML assertion.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *STS*.
2. Verify that the CardSpace attribute set is listed in the *Attribute sets* list.

   The CardSpace attribute set is a default set that ships with Access Manager. It contains all the claims that can be sent with an authentication card.
3. Click *Authentication Methods*.
4. Select a method, move it to the *Methods* list, then click *Apply*.

   The PasswordClass understands how to retrieve a name and password from a managed card. A method created from this class must be installed at the STS to provide authentication for the managed card. We recommend that you create a customized method from this class for CardSpace. For information on how to create methods, see Section 3.3, “Configuring Authentication Methods,” on page 123.

   If you are using the *Secure Name/Password - Form* method, you can select this method because it is created from PasswordClass.

   If you have installed a custom class that can retrieve CardSpace credentials and you have created a method for this class, you can select this method. For information on creating a custom authentication class, see *Novell Access Manager Developer Tools and Examples* (http://developer.novell.com/wiki/index.php/Novell_Access_Manager_Developer_Tools_and_Examples).
5. Click *Apply*, then click *Authentication Request*.

The options displayed allow you to select the format for the name identifier that is returned in the SAML assertion. The selected attribute sets (*Identity Servers* > *Edit* > *STS* > *Attribute Sets*) determine the values that are available for the formats.

6 Select a format and value.

If you select a format without a value type, a random one-time identifier is sent.

If no attributes are listed for the value type, you need to set up an attribute set. See Step 2.

None: Indicates that the SAML assertion does not contain a name identifier.

Unspecified: Specifies that the SAML assertion contains an unspecified name identifier. For the value, select the attribute that the relying party and the identity provider have agreed to use.

E-mail: Specifies that the SAML assertion contains the user's e-mail address for the name identifier. For the value, select an e-mail attribute.

X509: Specifies that the SAML assertion contains an X.509 certificate for the name identifier. For the value, select an X.509 attribute.

7 Click *Apply*, then restart the Identity Server:

7a On the Identity Servers page, select the server, then click *Stop* > *OK*.

7b When the health turns red, select the server, then click *Start*.

8.5.3 Creating a Managed Card Template

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Card Space* > *Managed Card Templates* > *New*, then fill in the following fields:

Name: Specify a display name for the template.

Description: Specify the text to be displayed on the card. This can contain information about how the card can be used or the type of resource that can be accessed with the card.

Image: Specify the image to be displayed on the card. Select the image from the drop-down list. To add an image to the list, click *Select local image*.

Require Identification of Relying Party in Security Token: Select this option to require the relying party to provide identification when it requests a security token.

Allow Users to Back a Managed Card Using a Personal Card: Select this option if you want to allow users to back a managed card with a personal card.

- When a managed card is backed by a personal card, the user enters the required credentials once, and thereafter only the card is needed for authentication.
- When a managed card is not backed by a personal card, the user must always enter the required credentials on authentication.

When the *Allow User to Back a Managed Card Using a Personal Card* option is selected, the user is presented with the option to back the managed card with a personal card. When it is not selected, the option to back the managed card with a personal card is removed from the user interface.

2 Click *Next*, then fill in the following fields:

Attribute set: From the list of available sets, select an attribute set. A default attribute set, named CardSpace, is available for CardSpace claims.

Selected claims: From the list of available claims, select the attributes for the managed card and move them to the list of selected claims.

Do not remove the *Personal Private Identifier* claim.

3. Click *Finish*.
4. Update the Identity Server.

8.6 Using CardSpace Cards for Authentication to Access Gateway Protected Resources

The protected resources on an Access Gateway are designed to rely on contracts for authentication. The CardSpace protocol uses cards for authentication. Therefore, to use the CardSpace protocol as the authentication authority for protected resources, you need to associate an authentication card profile with the authentication contract you are using for the protected resources.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts*.
2. Click the name of the contract you are using for protected resources.
3. Verify that the *Satisfiable by External Provider* option is enabled, then click *Authentication Card*.
4. Disable the *Show Card* option, then click *OK*.
5. Click *CardSpace* > *Authentication Card*, then in the *Profiles* section, select the profile you want to use with protected resources.

   If you select a profile that is configured only for a personal card, the user must supply a personal card to log in.

   If you select a profile that is configured for a managed card, the user can supply a managed card to log in.
6. Click *User Identification*, then configure the following fields:

   Satisfies contract: Select the contract that is used by the protected resource.

   Allow federation: Select this option so that the personal private identifier of the card can be associated with a user in the Identity Server's user store.

   Authenticate: Select this method for federation.
7. Click *OK* twice, then update the Identity Server.
8. (Optional) Verify the configuration by requesting access to a protected resource configured to use the contract you have enabled for CardSpace.

8.7 Managing CardSpace Trusted Providers

A trusted provider is an issuer of authentication tokens that you want to strongly trust. The provider has given you its issuer ID and its public key for the signing certificate. Tokens issued from this trusted provider are validated by using the public key certificate.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Trusted Providers*.
2. Select from the following actions:

   New: Launches the Create Trusted Identity Provider Wizard. See Section 8.7.1, "CardSpace Identity Provider Wizard," on page 247 for more information.

   Delete: Allows you to delete the selected identity provider.

   Enable: Enables the selected identity provider.

Disable: Disables the selected identity provider. When the provider is disabled, the server does not load the definition. However, the definition is not deleted.

3 Click *OK*, then update the Identity Server if you modified the configuration.

8.7.1 CardSpace Identity Provider Wizard

The CardSpace Wizard allows you to create a new identity provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Trusted Providers* > *New*.

2 Configure the following fields:

Name: Specify a display name for the provider. This name appears in the list of trusted providers that you can select for an authentication card profile.

Source: Specifies that the Provider ID is entered manually.

Provider ID: Specify the issuer ID of the trusted provider. For an Identity Server cluster, the issuer ID is the base URL of the Identity Server plus the following path:

```
/sts/services/Trust
```

For example, if the base URL is `https://test.lab.novell.com:8443/nidp`, the Provider ID is the following value:

```
https://test.lab.novell.com:8443/nidp/sts/services/Trust
```

Identity Provider: Specify the signing certificate of the Identity Server. You need to export the public key certificate to a file and make it available so that you can browse to the location of the file.

3 Click *Next*, then click *Finish* on the certificate page.

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

8.7.2 Renaming the CardSpace Provider

Use the CardSpace page to modify the display name of the identity provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *[Name of Identity Provider]*.

2 To modify the name, specify a new display name for the trusted provider in the *Name* text box.

This name appears in the list of trusted providers that you can select for an authentication card profile.

3 Click *OK* twice, then update the Identity Server.

8.7.3 Updating the Metadata of the CardSpace Provider

Use the Metadata page to edit the Provider ID and to reimport the signing certificate.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *[Name of Identity Provider]* > *Metadata*.

2 Verify that the ID value matches the provider ID.

3 Click *Edit*.

4 Modify the following fields as required:

Provider ID: Modify or specify a new issuer ID for the trusted provider.

Signing Certificate: Click *Browse* to find the signing certificate and import it.

5 Click *OK* twice, then update the Identity Server.

8.8 Managing Card Templates

You create managed card templates when you want the Identity Server to act as an identity provider. Users can then use the templates to create managed cards and use the cards to log into the Identity Server.

When a user uses a managed card, certain information about the managed card is stored on the user's computer: the card name, the date that the card was installed, a "valid-through" date, and a history of the sites where this card was used.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Managed Card Templates*.

The table displays the following information about the templates you have created.

Name: A list of the managed card templates that can be modified. To modify the template, click the name.

Description: The description you have provided for the template. This is an optional configuration field, so it might be blank.

2 Select from the following actions:

New: To create a new managed card template, click *New*. For configuration details, see Section 8.8.1, "General Template Details," on page 248.

Delete: To delete a managed card template, select the template, then click *Delete*. To delete all templates, click the *Name* check box, then click *Delete*.

3 Click *OK* twice, then update the Identity Server if you have modified the configuration.

8.8.1 General Template Details

Use the Managed Card page to create a new template or to modify the general details of an existing template.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Managed Card Templates* > *New or [Name of Card]*.

2 Configure the following fields:

Name: Specify a display name for the template.

Description: Specify the text to be displayed on the card. This can contain information about how the card can be used or the type of resource that can be accessed with the card.

Image: Specify the image to be displayed on the card. Select the image from the drop-down list. To add an image to the list, click *Select local image*.

Require Identification of Relying Party in Security Token: Select this option to require the relying party to provide identification when it requests a security token for the user that is using the card to establish authentication credentials.

Allow Users to Back a Managed Card Using a Personal Card: When this option is selected, the user is presented with the option to back the managed card with a personal card. When this option is not selected, the option to back the managed card with a personal card is removed from the user interface.

- When a managed card is backed by a personal card, the user enters the required credentials once, and thereafter only the card is needed for authentication.
- When a managed card is not backed by a personal card, the user must always enter the required credentials on authentication.

3 Select one of the following actions:

- To configure the attributes, continue with Section 8.8.2, "Template Attributes," on page 249.
- To save your changes, click *OK* twice, then update the Identity Server.

8.8.2 Template Attributes

Use the Attribute page to select the claims that are available on the managed card.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Managed Card Templates* > *[Name of Card]* > *Attributes*.

2 Configure the following fields:

Attribute set: From the list of available sets, select the default CardSpace set or the set that you have created for CardSpace claims. To create a new attribute set, select *New Attribute Set*.

If the set you have created for CardSpace is not listed, you need to configure the STS to use the set. Click *Identity Servers* > *Edit* > *STS* > *Attribute Sets* to manage the claims that are available.

Selected claims: From the list of *Available claims*, select the attributes for the managed card and move them to the list of *Selected claims*.

3 Click *OK* if you are modifying a template, or click *Finish* if you are creating a template.

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

8.9 Configuring Authentication Cards

Use the Authentication Card page to manage the card assigned to a CardSpace authentication card.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Authentication Card*.

2 Configure the following fields:

ID: (Optional) Specify an alphanumeric value that identifies the card. If you need to reference this card outside of the user interface, you must specify a value here. If you do not assign a value, the Identity Server creates one for its internal use.

Text: Specify the text that is displayed on the card to the user.

Image: Specify the image to be displayed on the card. Select the image from the drop-down list. To add an image to the list, click *Select local image*.

Show Card: Determine whether the card is shown to the user, which allows the user to select and use the card for authentication. If this option is not selected, the card is only used when a service provider requests the card.

3 Select from the following actions:

New: To create a new profile, click *New.* For configuration information, see Section 8.9.1, "Configuring the General Details of a Card Profile," on page 250.

A card profile allows you to provide different authentication options for the same card. When creating a profile, you select the type of provider that can issue the card, the claims that must have values in the card, and the method that is used to identify the user.

To create an authentication card profile, you must have at least one attribute set available that contains the claims you want to use for the card. To create an attribute set, click *Identity Servers* > *Shared Settings* > *Attributes Sets*.

Modify: To modify an existing profile, click the name of the profile. For configuration information, see Section 8.9.1, "Configuring the General Details of a Card Profile," on page 250.

Make Default: To make a profile the default, select the profile, then click *Make Default*.

Delete: To delete a profile, select the profile, then click *Delete*.

4 Click *OK*, then update the Identity Server if you have changed the configuration.

8.9.1 Configuring the General Details of a Card Profile

Use the Card Profile page to create a new card profile or to modify an existing profile.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Authentication Card Profiles* > *New [Name of Profile]*.

2 Configure the following fields:

Name: Specify a display name for the profile.

ID: (Optional) Specify an alphanumeric value (no spaces) that identifies the card. If you need to reference this card outside of the user interface, you must specify a value here. If you do not assign a value, the Identity Server creates one for its internal use.

Text: Specify the text that is displayed on the card to the user.

Issuer: From the drop-down list, select the issuer for the card.

- **Any Trusted or Untrusted Provider or Personal Card:** Specifies that the card can be either a personal card or a managed card from both trusted and untrusted providers.
- **Personal Card:** Specifies that the card must be a personal card.
- **Any Trusted Provider or Personal Card:** Specifies that the card can be either a personal card or a managed card from any trusted provider.
- **<Provider Name>:** Specifies that the card must be a managed card from the specified provider. To add a trusted provider, click *Identity Servers* > *Edit* > *CardSpace* > *Trusted Providers* > *New*.

Token Type: Indicates that the authentication credential is a SAML 1.1 token.

3 Select one of the following actions:

- If you are creating a profile, click *Next*. Continue with Section 8.9.2, "Configuring Attribute Claims," on page 251.
- If you have finished modifying the profile, click *OK* twice, then update the Identity Server.

- To modify the profile attributes, click *Attributes*. Continue with Section 8.9.2, “Configuring Attribute Claims,” on page 251.
- To modify the user identification methods, click *User Identification*. Continue with Section 8.9.3, “Configuring User Identification,” on page 251.

8.9.2 Configuring Attribute Claims

Use the Attributes page to specify the attributes (claims) that must have values.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Authentication Card Profiles* > *[Name of Profile]* > *Attributes*.
2. Configure the following fields:

   Attribute Set: From the drop-down list, select the attribute set from which you want to select required and optional attributes. These attributes must match the claims that have been defined for personal cards. If you need to create an attribute set, select *New Attribute Set*. See Section 6.1, “Configuring Attribute Sets,” on page 177.

   Required Attributes: From the list of available attributes, select an attribute and move it to the *Required Attribute* list. If the managed card is going to be backed by a personal card, make sure the *Personal Private Identifier* attribute is selected.

   Optional Attributes: From the list of available attributes, select an attribute and move it to the *Optional Attribute* list.
3. Select one of the following actions:
   - If you are creating a profile, click *Next*. Continue with Section 8.9.3, “Configuring User Identification,” on page 251.
   - If you have finished modifying the profile, click *OK* twice, then update the Identity Server.
   - To modify the user identification methods, click *User Identification*. Continue with Section 8.9.3, “Configuring User Identification,” on page 251).

8.9.3 Configuring User Identification

Use this page to specify the user identification methods. The options on this page determine whether the user can use the card for single sign-on.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Authentication Card* > *[Name of Profile]* > *User Identification*.
2. Configure the following fields:

   Satisfied Contracts: From the list of available contracts, select a contract and move it to the *Satisfied Contract* list. Select one or more.

   If you are using CardSpace to allow access to Access Gateway protected resources, you must ensure that all contracts specified for a protected resource are satisfied by an authentication profile.

   Allow Federation: Select this option to enable account federation. Enabling this option assumes that a user account exists at the provider or that a method is provided to create an account that can be associated with the user on subsequent logins. If you do not use this feature, authentication is permitted but is not associated with a particular user account.

3 Select one of the following user identification methods for associating the accounts:

- **Do nothing:** Allows the user to authenticate without creating an association with a user account. This option cannot be used when federation is enabled.
- **Authenticate:** Select this option when you want to use login credentials. This option prompts the user to log in to the service provider.
  - **Allow 'Provisioning':** Select this option to allow users to create an account when they have no account on the service provider.

    This option requires that you specify a user provisioning method.
- **Provision Account:** Select this option when the users on the identity provider do not have accounts on the service provider. This option allows the service provider to trust any user that has authenticated to the trusted identity provider.

  This option requires that you specify a user provisioning method.
- **Attribute matching:** Select this option when you want to use attributes to match an identity server account with a service provider account. This option requires that you specify a user matching method.
  - **Prompt for password on successful match:** Select this option to prompt the user for a password when the user's name is matched to an account, to ensure that the account matches.

4 (Conditional) If you selected a user identification method that requires a matching method or a provision setting, configure the required method.

Provisioning Settings: Allows you to select or create a user provisioning method. See Section 11.3, "Defining the User Provisioning Method," on page 286. For user provisioning error messages, see Section 11.4, "User Provisioning Error Messages," on page 290.

Attribute Matching Settings: Allows you to select or create a user matching method. See Configuring the Attribute Matching Method for Liberty or SAML 2.0.

5 If you are creating a new profile, click *Finish*, or if you are modifying a profile, click *OK*.

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

8.10 Cleaning Up Identities

Use the Configuration page to manage time limits for identity cleanup.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *CardSpace* > *Configuration*.

2 Configure the following fields:

Maximum Age (for Unused Identities): Specifies how long an account can remain inactive before the account is defederated. The default limit is 90 days. Specify a value from 0 to 365 days.

Maximum Age (for Managed Cards Backed by Personal Cards): Specifies how long a managed card, backed by a personal card, can remain valid without the token being refreshed. When this limit is reached, the managed card is deleted. The default limit is 90 days. Specify a value from 0 to 365 days.

3 Click *OK*, then update the Identity Server if you have changed the configuration.

Configuring STS

9

The STS (Security Token Service) is used to process authentication requests received at the Identity Server for both the CardSpace and the WS Federation protocols.

9.1 Configuring STS Attribute Sets

Use the Attribute Set page to select the attribute set or sets that contain attributes the STS can provide to a relying party. An attribute set must be created before you can select it.

When creating an attribute set for the STS, you need to know which protocol you are going to use for the attribute set (CardSpace or WS Federation) and select the attributes and namespace appropriate for the protocol.

1. In the Administrations Console, click *Devices* > *Identity Servers* > *Edit* > *STS* > *Attribute Sets*.
2. To select a set, move the set from the *Available attribute sets* list to the *Attribute sets* list.

   CardSpace: A CardSpace set uses the `http://schemas.xmlsoap.org/ws/2005/05/identity/claims` namespace. A CardSpace attribute set has been created that can be used as is or modified to match claims you want to share. For more information about CardSpace claims, see Understanding Personal Information Cards (http://msdn.microsoft.com/en-us/library/aa347717.aspx)

   To modify this default set, click *Identity Servers* > *Shared Settings* > *Attribute Sets*, then return to this page.

   WS Federation: There is no default attribute set for WS Federation. For information on how to create the set, see Section 10.4.2, “Configuring the Attributes Obtained at Authentication,” on page 274 and Section 10.5.2, “Configuring the Attributes Sent with Authentication,” on page 277.
3. Click *OK*, then update the Identity Server if you have changed the configuration.

9.2 Configuring Authentication Methods

Use the Authentication Methods page to select the methods that can be used for authentication at the STS for CardSpace. The methods determine the credentials the user must supply for authentication and the user store that is used to verify the credentials. The WS Federation protocol does not use methods for authentication.

1. In the Administrations Console, click *Devices* > *Identity Servers* > *Edit* > *STS* > *Authentication Methods*.
2. To enable a method, move the method from the *Available methods* list to the *Methods* list.

All the methods that you have defined for the Identity Server appear in the *Available methods* list, but the only default method that works is the Secure Name/Password-Form method. It has been extended so that it knows how to extract name and password information from a managed card that is not backed by a personal card. You can use the Secure Name/Password-Form class to create additional methods for specific user stores.

You can also create a custom method, if required. For information, see Novell Access Manager Developer Tools and Examples (http://developer.novell.com/wiki/index.php/Novell_Access_Manager_Developer_Tools_and_Examples).

3 Click *OK*, then update the Identity Server if you have changed the configuration.

9.3 Configuring the Authentication Request

Use the Authentication Request page to select the format for the name identifier that is returned in the SAML assertion. The selected attribute sets (*Identity Servers* > *Edit* > *STS* > *Attribute Sets*) determine the values that are available for the formats. If you select a format but do not specify a value, a unique value is generated.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *STS* > *Authentication Request*.

2 Select one of the following:

None: Indicates that the SAML assertion does not contain a name identifier.

Unspecified: Specifies that the SAML assertion contains an unspecified name identifier. For the value, select the attribute that the relying party and the identity provider have agreed to use.

E-mail: Specifies that the SAML assertion contains the user's e-mail address for the name identifier. For the value, select an e-mail attribute.

X509: Specifies that the SAML assertion contains an X.509 certificate for the name identifier. For the value, select an X.509 attribute.

3 Click *OK*, then update the Identity Server if you have changed the configuration.

Configuring WS Federation

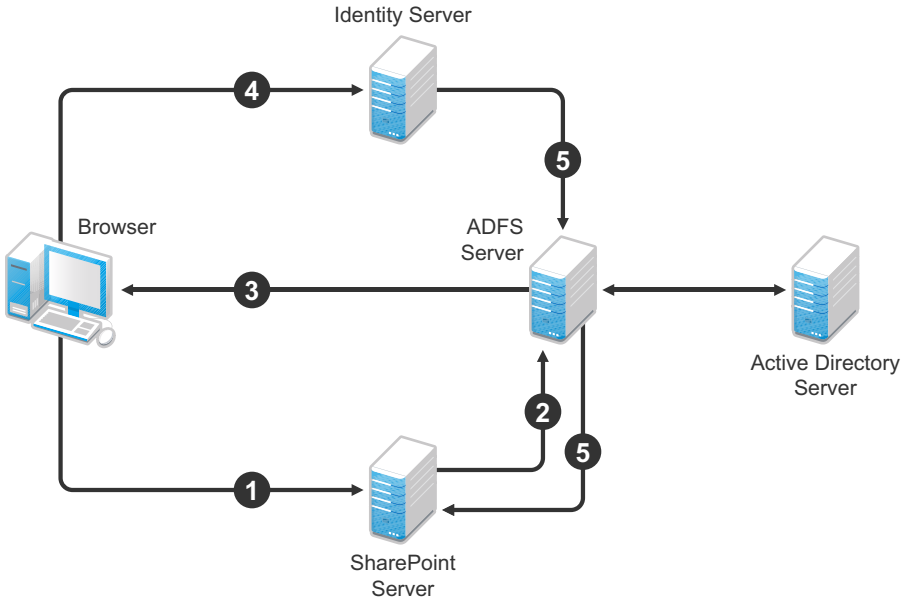
10

The first two topics in this section describe two different methods for setting up federation with a SharePoint server. The next sections describe how you can manage and modify WS Federation providers.

10.1 Using the Identity Server as an Identity Provider for ADFS

The Identity Server can provide authentication for resources protected by an Active Directory Federation Services (ADFS) server. This allows the Identity Server to provide single sign-on to Access Manager resources and ADFS resources, such as a SharePoint server. Figure 10-1 illustrates this configuration.

Figure 10-1 *Accessing SharePoint Resources with an Identity Server*



In this scenario, the following exchanges occur:

1. The user requests access to a SharePoint server protected by the ADFS server.
2. The resource sends an authentication request to the ADFS server.

3. The ADFS server, which has been configured to use the Identity Server as an identity provider, gives the user the option of logging in to the Identity Server.
4. The user logs in to the Identity Server and is provided a token that is sent to the ADFS server and satisfies the request of the resource.
5. The user is allowed to access the resource.

The following section describe how to configure your servers for this scenario:

10.1.1 Configuring the Identity Server

Prerequisites

- You have set up the Active Directory Federation Services, Active Directory, and SharePoint servers and the client as described in the ADFS guide from Microsoft. See the “Step-by-Step Guide for Active Directory Federation Services” (http://go.microsoft.com/fwlink/?linkid=49531).
- You have set up the Novell Access Manager 3.1 system with a site configuration that is using SSL in the Identity Server's base URL. See “Enabling SSL Communication” in the *Novell Access Manager 3.1 SP2 Setup Guide*.

Creating a New Authentication Contract

The Microsoft ADFS server rejects the contract URI names of the default Access Manager contracts, which have a URI format of `secure/name/password/uri`. The ADFS server expects the URI to look like a URL.

We suggest that you use the following format for the URI of all contracts that you want to use with the ADFS server:

```
<baseurl>/name/password/uri
```

If the DNS name of your Identity Server is `idp-50.amlab.net`, the URI would have the following format:

```
https://idp-50.amlab.net:8443/nidp/name/password/uri
```

This URL doesn't resolve to anything because the Identity Server interprets it as a contract URI and not a URL.

To create a new authentication contract:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local* > *Contracts*.
2. Click *New*, then fill in the following fields:

   Display name: Specify a name, for example WS-Fed Contract.

   URI: Specify a URI, for example `https://idp-50.amlab.net:8443/nidp/name/password/uri`.

   Satisfiable by External Provider: Enable this option. The ADFS server needs to satisfy this contract.
3. Move *Name/Password – Form* to the *Methods* list.
4. Click *Next*, then fill in the following fields:

   ID: Leave this field blank. You only need to supply a value when you want a reference that you can use externally.

   Text: Specify a description that is available to the user when the user mouses over the card.

   Image: Select an image, such as *Form Auth Username Password*. This is the default image for the Name/Password - Form contract.

   Show Card: Enable this option so that the card can be presented to the user as a login option.
5. Click *Finish*.
6. Continue with "Setting the WS-Fed Contract to Be the Default Contract" on page 257.

Setting the WS-Fed Contract to Be the Default Contract

It is not possible to specify the contract to request from the ADFS service provider to the Identity Server. You must either set the contract for WS-Fed to be the default, or have your users remember to click that contract every time.

1. On the Local page of the Identity Server, click *Defaults*.
2. For the *Authentication Contract* option, select the WS-Fed Contract.
3. Click *Apply*.
4. Continue with "Enabling the STS and WS Federation Protocols" on page 257.

Enabling the STS and WS Federation Protocols

Access Manager ships with only SAML 1.1, Liberty, and SAML 2.0 enabled by default. In order to use the WS Federation protocol, you must enable it on the Identity Server. Because the WS Federation Protocol uses the STS (Secure Token Service) protocol, STS must also be enabled.

1. Click the *General* tab.
2. In the *Enabled Protocols* section, select the STS and WS Federation protocols.
3. Click *OK*.

4 Update the Identity Server.

5 Continue with "Creating an Attribute Set for WS Federation" on page 258.

Creating an Attribute Set for WS Federation

The CardSpace attribute set is not in the correct namespace for WS Federation. The WS Federation namespace is `http://schemas.xmlsoap.org/claims`. Also, CardSpace has a defined set of claims. With WS Federation, you need to decide which attributes you want to share during authentication. This scenario uses the LDAP mail attribute and the All Roles attribute.

1 On the Identity Servers page, click *Shared Settings*.

2 To create a new attribute set, click *New*, then fill in the following fields:

Set Name: Specify a name that identifies the purpose of the set, for example, wsfed_attributes.

Select set to use as template: Select *None*.

3 Click *Next*.

4 To add a mapping for the mail attribute:

4a Click *New*.

4b Fill in the following fields:

Local attribute: Select *LDAP Attribute:mail [LDAP Attribute Profile]*.

Remote attribute: Specify *emailAddress*. This is the attribute that this scenario uses for user identification.

Remote nanespace: Select the radio button by the text box, then specify the following namespace:

`http://schemas.xmlsoap.org/claims`

4c Click *OK*.

5 To add a mapping for the All Roles attribute:

5a Click *New*.

5b Fill in the following fields:

Local attribute: Select *All Roles*.

Remote attribute: Specify *group*. This is the name of the attribute that is used to share roles.

Remote nanespace: Select the radio button by the text box, then specify the following namespace:

`http://schemas.xmlsoap.org/claims`

5c Click *OK*.

6 Click *Finish*.

7 Continue with "Enabling the Attribute Set" on page 258.

Enabling the Attribute Set

Because the WS Federation protocol uses STS, you must enable the attribute set for STS in order to use it in an WS Federation relationship.

1 On the Identity Servers page, click *Servers* > *Edit* > *STS*.

2 Move the WS Federation attribute set to the *Attribute set* list.

3 Select the WS Federation attribute set and use the up-arrow to make it first in the *Attribute set* list.

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

Creating a WS Federation Service Provider

In order to establish a trusted relationship with the ADFS server, you need to set up the Trey Research site as a service provider. The trusted relationship allows the service provider to trust the Identity Server for user authentication credentials.

Trey Research is the default name for the ADFS resource server. If you have used another name, substitute it when following these instructions. To create a service provider, you need to know the following about the ADFS resource server.

Table 10-1 *ADFS Resource Server Information*

What You Need to Know	Default Value and Description
Provider ID	**Default Value:** `urn:federation:treyresearch` This is the value that the ADFS server provides to the Identity Server in the realm parameter of the query string. This value is specified in the Properties of the Trust Policy page on the ADFS server. The parameter label is *Federation Service URI*.
Sign-on URL	**Default Value:** `https://adfsresource.treyresearch.net/adfs/ls/` This is the value that the identity provider redirects the user to after login. Although it is listed as optional, and is optional between two Novell Identity Servers, the ADFS server doesn't send this value to the identity provider. It is required when setting up a trusted relationship between an ADFS server and a Novell Identity Server. This URL is listed in the Properties of the Trust Policy page on the ADFS server. The parameter label is *Federation Services endpoint URL*.
Logout URL	**Default Value:** `https://adfsresource.treyresearch.net/adfs/ls/` This parameter is optional. If it is specified, the user is logged out of the ADFS server and the Identity Server.
Signing Certificate	This is the certificate that the ADFS server uses for signing. You need to export it from the ADFS server. It can be retrieved from the properties of the *Trust Policy* on the ADFS Server on the *Verification Certificates* tab. This certificate is a self-signed certificate that you generated when following the Active Directory step-by-step guide.

To create a service provider configuration:

1 On the Identity Servers page, click *Edit* > *WS Federation*.

2 Click *New* > *Service Provider*, then fill in the following fields:

Name: Specify a name that identifies the service provider, such as `TreyResearch`.

Provider ID: Specify the provider ID of the ADFS server. The default value is `urn:federation:treyresearch`.

Sign-on URL: Specify the URL that the user is redirected to after login. The default value is `https://adfsresource.treyresearch.net/adfs/ls/`.

Logout URL: (Optional) Specify the URL that the user can use for logging out. The default value is `https://adfsresource.treyresearch.net/adfs/ls`.

Service Provider: Specify the path to the signing certificate of the ADFS server.

3. Click *Next*, confirm the certificate, then click *Finish*.
4. Continue with "Configuring the Name Identifier Format" on page 260.

Configuring the Name Identifier Format

The Unspecified Name Identifier format is the default for a newly created WS Federation service provider, but this name identifier format doesn't work with the ADFS federation server. Additionally, some Group Claims (Adatum ClaimApp Claim and Adatum TokenApp Claim) must be satisfied in order to gain access to the SharePoint server.

1. On the WS Federation page, click the name of the TreyResearch service provider.
2. Click *Attributes*, then fill in the following fields:

   Attribute set: Select the WS Federation attribute set you created.

   Send with authentication: Move the All Roles attribute to the *Send with authentication* list.
3. Click *Apply*, then click *Authentication Response*.
4. Select *E-mail* for the Name Identifier Format.
5. Select *LDAP Attribute:mail [LDAP Attribute Profile]* as the value for the e-mail identifier.
6. Click *OK* twice, then update the Identity Server.
7. Continue with "Setting Up Roles for ClaimApp and TokenApp Claims" on page 260.

Setting Up Roles for ClaimApp and TokenApp Claims

When users access resources on the ADFS server, they need to have two roles assigned: a ClaimApp role and a TokenApp role. The following steps explain how to create these two roles so that they are assigned to all users that log in to the Identity Server.

1. On the Identity Servers page, click *Edit* > *Roles* > *Manage Policies*.
2. Click *New*, specify a name for the policy, select *Identity Server: Roles*, then click *OK*.
3. On the Rule 1 page, leave Condition Group 1 blank.

   With no conditions to match, this rule matches all authenticated users.
4. In the *Actions* section, click *New* > *Activate Role*.
5. In the text box, specify *ClaimApp*.
6. In the *Actions* section, click *New* > *Activate Role*.
7. In the text box, specify *TokenApp*.
8. Click *OK* twice, then click *Apply Changes*.
9. Click *Close*.
10. On the Roles page, select the role policy you just created, then click *Enable*.

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

12 Continue with "Importing the ADFS Signing Certificate into the NIDP-Truststore" on page 261.

Importing the ADFS Signing Certificate into the NIDP-Truststore

The Novell Identity Provider (NIDP) must have the trusted root of the ADFS signing certificate (or the certificate itself) listed in its Trust Store, as well as specified in the relationship. This is because most ADFS signing certificates are part of a certificate chain, and the certificate that goes into the metadata is not the same as the trusted root of that certificate. However, because the Active Directory step-by-step guide uses self-signed certificates for signing, it is the same certificate in both the Trust Store and in the relationship.

To import the ADFS signing certificate's trusted root (or the certificate itself) into the NIDP-Truststore:

1 On the Identity Servers page, click *Edit* > *Security* > *NIDP Trust Store*.

2 Click *Add*.

3 Next to the *Trusted Root(s)* field, click the *Select Trusted Root(s)* icon.

This adds the trusted root of the ADFS signing certificate to the Trust Store.

4 On the Select Trusted Roots page, select the trusted root or certificate that you want to import, then click *Add Trusted Roots to Trust Stores*.

If there is no trusted root or certificate in the list, click *Import*. This enables you to import a trusted root or certificate.

5 Next to the *Trust store(s)* field, click the *Select Keystore* icon.

6 Select the trust stores where you want to add the trusted root or certificate, then click *OK* twice.

7 Update the Identity Server so that the changes can take effect.

This finishes the configuration that must be done on the Identity Server for the Identity Server to trust the ADFS server. The ADFS server must be configured to trust the Identity Server. Continue with Section 10.1.2, "Configuring the ADFS Server," on page 261.

10.1.2 Configuring the ADFS Server

The following tasks must be completed on the Trey Research server (adfsresouce.treyresearch.net) to establish trust with the Novell Identity Server.

- "Enabling E-mail as a Claim Type" on page 262
- "Creating an Account Partners Configuration" on page 262
- "Enabling ClaimApp and TokenApp Claims" on page 263
- "Disabling CRL Checking" on page 263

Enabling E-mail as a Claim Type

There are three types of claims for identity that can be enabled on an ADFS server. They are Common Name, E-mail, and User Principal Name. The ADFS step-by-step guide specifies that you do everything with a User Principal Name, which is an Active Directory convention. Although it could be given an e-mail name that looks the same, it is not. This scenario selects to use E-mail instead of Common Name because E-mail is a more common configuration.

1. From the Administrative Tools, open the Active Directory Federation Services tool.
2. Navigate to the *Organizational Claims* by clicking *Federation Service* > *Trust Policy* > *My Organization*.
3. Verify that E-mail is in this list. If it isn't, move it to the list.
4. Navigate to your Token-based Application and enable e-mail by right-clicking the application, editing the properties, and clicking the *Enabled* box.
5. Navigate to your Claims-aware Application and repeat the process.
6. Continue with "Creating an Account Partners Configuration" on page 262.

Creating an Account Partners Configuration

WS Federation requires a two-way trust relationship. Both the identity provider and the service provider must be configured to trust the other provider. This task sets up the trust between the ADFS server and the Identity Server.

1. In the Active Directory Federation Services console, navigate to the Account Partners by clicking *Federation Services* >*Trust Policy* > *Partner Organizations*.
2. Right-click *Partner Organizations*, then select *New* > *Account Partner*.
3. Supply the following information in the wizard:
   - You do not have an account partner policy file.
   - For the display name, specify the DNS name of the Identity Server.
   - For the *Federation Services URI,* specify the following:

     `https://<DNS_Name>:8443/nidp/wsfed/`

     Replace <*DNS_Name*> with the DNS name of the Identity Server.

     This URI is the base URL of your Identity Server with the addition of `/wsfed/` on the end.
   - For the *Federation Services endpoint URL*, specify the following:

     `https://<DNS_Name>:8443/nidp/wsfed/ep`

     Replace <*DNS_Name*> with the DNS name of the Identity Server.

     This URL is the base URL of your Identify Server with the addition of `/wsfed/ep` at the end.
   - For the verification certificate, import the trusted root of the signing certificate on your Identity Server.

     If you have not changed it, you need the Organizational CA certificate from your Administration Console. This is the trusted root for the test-signing certificate.
   - Select *Federated Web SSO*.

     The Identity Server is outside of any forest, so do not select *Forest Trust*.
   - Select the E-mail claim.

- Add the suffix that you will be using for your e-mail address.

  You need to have the e-mail end in a suffix that the ADFS server is expecting, such as @novell.com, which grants access to any user with that e-mail suffix.

4. Enable this account partner.
5. Finish the wizard.
6. Continue with "Enabling ClaimApp and TokenApp Claims" on page 263.

Enabling ClaimApp and TokenApp Claims

The Active Directory step-by-step guide sets up these roles to be used by the resources. You set them up to be sent in the All Roles attribute from the Identity Server. You must map these roles into the Adatum ClaimApp Claim and the Adatum TokenApp Claim.

1. In the Active Directory Federation Services console, click the account partner that you created for the Identity Server (see "Creating an Account Partners Configuration" on page 262).
2. Right click the account partner, then create a new *Incoming Group Claim Mapping* with the following values:

   Incoming group claim name: Specify *ClaimApp*.

   Organization group claim: Specify *Adatum ClaimApp Claim*.
3. Right-click the account partner, and create another *Incoming Group Claim Mapping* with the following values:

   Incoming group claim name: Specify *TokenApp*.

   Organization group claim: Specify *Adatum TokenApp Claim*.
4. Continue with "Disabling CRL Checking" on page 263.

Disabling CRL Checking

If you are using the Access Manager certificate authority as your trusted root for the signing certificate (test-signing certificate), there is no CRL information in that certificate. However, the ADFS has a mandatory requirement to do CRL checking on any certificate that they receive. For instructions on how to disable this checking, see "Turn CRL checking on or off" (http://go.microsoft.com/fwlink/?LinkId=68608).

Use the following tips as you follow these instructions.

- Create a file from the script contained at that link called `TpCrlChk.vbs`.
- Exit the Active Directory Federation Services console.

  If you do not exit the console, the console overwrites the changes made by the script file and CRL checking is not turned off.
- Run the command with the following syntax:

  ```
  Cscript TpCrlChk.vbs <location of ADFS>\TrustPolicy.xml "<service URI>"
  None
  ```

  Replace <*location of ADFS*> with the location of the ADFS `TrustPolicy.xml` file. The default location is `C:\ADFS\TrustPolicy.xml`.

  Replace <*service URI*> with the URI you specified in Step 3 on page 262. If the DNS name of your Identity Server is `idp-50.amlab.net`, replace it with `https://idp-50.amlab.ne:8443/nidp/wsfed/`.

Your command should look similar to the following:

```
Cscript TpCrlChk.vbs C:\ADFS\TrustPolicy.xml "https://idp-
50.amlab.net:8443/nidp/wsfed/" None
```

10.1.3 Logging In

1. In a browser on your client machine, enter the URL of the SharePoint server. For example:

   ```
   https://adfsweb.treyresearch.net/default.aspx
   ```

2. Select the IDP from the drop-down list of *home realm*, then submit the request.

   If you are not prompted for the realm, clear all cookies in the browser and try again.

3. Log in with a user at the Novell Identity Provider
4. Verify that you can access the SharePoint server.

   If you only see a page that says `Server Error in '/adfs' Application`, see "Turning On Logging on the ADFS server" on page 264 and follow the instructions in "Common Errors" on page 264.

10.1.4 Troubleshooting

Turning On Logging on the ADFS server

If you see the message `Server Error in '/adfs' Application` displayed in the client's browser, the best place to look for the cause is in the ADFS log file.

To turn on this log file:

1. In the Active Directory Federation Services console, right-click *Federation Service*, then click *Properties*.
2. Click the *Troubleshooting* tab, then enable everything on the page.
3. Click *OK*, then look for the file that is created in the path listed in the *Log files directory*.
4. Look in that file for reasons that the federation is failing.

   For an explanation of some of the common errors, see "Common Errors" on page 264.

Common Errors

[ERROR] SamlViolatesSaml:

Error parsing AuthenticationMethod: Invalid URI: The format of the URI could not be determined.

Cause: This is because the contract has the wrong format for its URI. The URI must start with `urn:` or `http://`. Change the contract and try again.

[ERROR] Saml contains an unknown NameIdentifierFormat:

Issuer=https://idp-51.amlab.net:8443/nidp/wsfed/; Format=urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

Cause: The name identifier format is set to unspecified, and it needs to be set to E-mail.

[ERROR] Saml contains an unknown Claim name/namespace:

Issuer=https://idp-51.amlab.net:8443/nidp/wsfed/;
Namespace=urn:oasis:names:tc:SAML:1.0:assertion; Name=emailaddress

Cause: The emailAddress attribute is not in the correct namespace for WSFed.

CRL Errors

- 2008-08-01T19:56:55 [WARNING] VerifyCertChain: Cert chain did not verify - error code was 0x80092012
- 2008-08-01T19:56:55 [ERROR] KeyInfo processing failed because the trusted certificate does not have a a valid certificate chain. Thumbprint = 09667EB26101A98F44034A3EBAAF9A3A09A0F327
- 2008-08-01T19:56:55 [WARNING] Failing signature verification because the KeyInfo section failed to produce a key.
- 2008-08-01T19:56:55 [WARNING] SAML token signature was not valid: AssertionID = idZ0KQH0kfjVK8kmKfv6YaVPglRNo

Cause: The CRL check isn't turned off. See “Disabling CRL Checking” on page 263.

[ERROR] EmailClaim.set_Email:

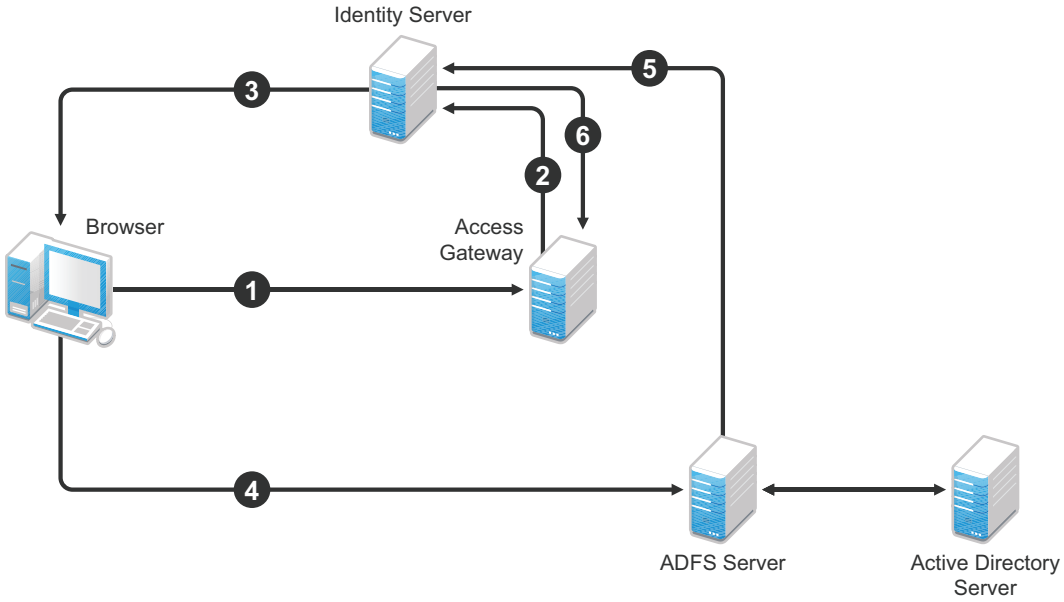
Email 'mPmNXOA8Rv+j16L1iNKn/4HVpfeJ3av1L9c0GQ==' has invalid format

Cause: The drop-down list next to E-mail in the identifier format was not changed from <Not Specified> to a value with a valid e-mail address in it.

10.2 Using the ADFS Server as an Identity Provider for an Access Manager Protected Resource

The Active Directory Federation Services server can be configured to provide authentication for a resource protected by Access Manager.

Figure 10-2 *Using an ADFS Server for Access Manager Authentication*



In this scenario, the following exchanges occur:

1. The user requests access to a resource protected by an Access Gateway.
2. The resource sends an authentication request to the Novell Identity Server.
3. The Identity Server is configured to trust an Active Directory Federation Services server and gives the user the option of logging in at the Active Directory Federation Services server.
4. The user logs into the Active Directory Federation Services server and is provided a token
5. The token is sent to the Identity Server.
6. The token satisfies the authentication requirements of the resource, so the user is allowed to access the resource.

The following sections describe how to configure this scenario.

10.2.1 Configuring the Identity Server as a Service Provider

Prerequisites

- You have set up the Active Directory Federation Services, Active Directory, and SharePoint servers and the client as described in the ADFS guide from Microsoft. See the "Step-by-Step Guide for Active Directory Federation Services" (http://go.microsoft.com/fwlink/?linkid=49531).
- You have set up the Novell Access Manager 3.1 system with a site configuration that is using SSL in the Identity Server's base URL. See "Enabling SSL Communication" in the *Novell Access Manager 3.1 SP2 Setup Guide*.
- Enable the Liberty Personal Profile.

  In the Administration Console, click *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*. Select the *Personal Profile*, then click *Enable* > *Apply*. Update the Identity Server.

Enabling the STS and WS Federation Protocols

Access Manager ships with only SAML 1.1, Liberty, and SAML 2.0 enabled by default. In order to use the WS Federation protocol, it must be enabled on the Identity Server. Because the WS Federation Protocol uses the STS (Secure Token Service) protocol, STS must also be enabled.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit*.
2. In the *Enabled Protocols* section of the General Configuration page, enable the STS and WS Federation protocols.
3. Click *OK*.
4. Update the Identity Server.
5. Continue with "Creating a WS Federation Identity Provider" on page 267.

Creating a WS Federation Identity Provider

In order to have a trust relationship, you need to set up the Adatum site (adfsaccount.adatum.com) as an identity provider for the Identity Server.

Adatum is the default name for the identity provider. If you have used another name, substitute it when following these instructions. To create an identity provider, you need to know the following information about the Adatum site:

Table 10-2 *Adatum Values*

What You Need to Know	Default Value and Description
Provider ID	**Default Value:** `urn:federation:adatum` The ADFS server provides this value to the service provider in the realm parameter in the assertion. You set this value in the *Properties* of the Trust Policy on the ADFS server. The label is *Federation Service URI*.

What You Need to Know	Default Value and Description
Sign-on URL	**Default Value:** `https://adfsaccount.adatum.com/adfs/ls/` The service provider uses this value to redirect the user for login. This URL is listed in the *Properties* of the Trust Policy on the ADFS server. The label is *Federation Services endpoint URL*.
Logout URL	**Default Value:** `https://adfsresource.treyresearch.net/adfs/ls/` The ADFS server makes no distinction between the login and logout URL. Access Manager has separate URLs for login and logout, but from a Novell Identity Server to an ADFS server, they are the same.
Signing Certificate	This is the certificate that the ADFS server uses for signing. You need to export it from the ADFS server. It can be retrieved from the properties of the *Trust Policy* on the ADFS Server on the *Verification Certificates* tab. This certificate is a self-signed certificate that you generated when following the step-by-step guide.

To create an identity provider:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation*.
2. On the WS Federation page, click *New*, select *Identity Provider*, then fill in the following fields:

   Name: Specify a name that identifies the identity provider, such as `Adatum`.

   Provider ID: Specify the federation service URI of the identity provider, for example `urn:federation:adatum`.

   Sign-on URL: Specify the URL for logging in, such as `https://adfsaccount.adatum.com/adfs/ls/`.

   Logout URL: Specify the URL for logging out, such as `https://adfsresource.treyresearch.net/adfs/ls/`

   Identity Provider: Specify the path to the signing certificate of the ADFS server.
3. Confirm the certificate, then click *Next*.
4. For the authentication card, specify the following values:

   ID: Leave this field blank.

   Text: Specify a description that is available to the user when the user mouses over the card.

   Image: Select an image, such as *Customizable*, or any other image.

   Show Card: Enable this option so that the card can be presented to the user as a login option.
5. Click *Finish*.
6. Continue with "Modifying the User Identification Specification" on page 268.

Modifying the User Identification Specification

The default settings for user identification are set to do nothing. The user can authenticated but the user is not identified as a local user on the system. This is not the scenario we are configuring. We want the user to be identified on the local system. Additionally, we want to specify which contract

on the Access Gateway is satisfied with this identification. If a contract is not specified, the Access Gateway resources must be configured to use the *Any Contract* option, which is not a typical configuration.

1. On the WS Federation page, click the name of the Adatum identity provider configuration.
2. Click *User Identification*.
3. For *Satisfies contract*, select *Name/Password – Form*.
4. Select *Allow federation*.
5. For the *User Identification Method*, select *Authenticate*.
6. Click *OK* twice.
7. Update the Identity Provider.
8. Continue with "Importing the ADFS Signing Certificate into the NIDP-Truststore" on page 269.

Importing the ADFS Signing Certificate into the NIDP-Truststore

The Identity Server must have the trusted root of the ADFS signing certificate (or the certificate itself) listed in its trust store, as well as specified in the relationship. This is because most ADFS signing certificates have a chain, and the certificate that goes into the metadata is not the same as the trusted root of that certificate. However, because the Active Directory step-by-step guide uses self-signed certificates for signing, it is the same certificate in both the trust store and in the relationship.

To import the ADFS signing certificate's trusted root (or the certificate itself) into the NIDP-Truststore:

1. On the Identity Servers page, click *Edit* > *Security* > *NIDP Trust Store*.
2. Click *Add*.
3. Next to the *Trusted Root(s)* field, click the *Select Trusted Root(s)* icon.

   This adds the trusted root of the ADFS signing certificate to the Trust Store.
4. On the Select Trusted Roots page, select the trusted root or certificate that you want to import, then click *Add Trusted Roots to Trust Stores*.

   If there is no trusted root or certificate in the list, click *Import*. This enables you to import a trusted root or certificate.
5. Next to the *Trust store(s)* field, click the *Select Keystore* icon.
6. Select the trust stores where you want to add the trusted root or certificate, then click *OK* twice.
7. Update the Identity Server so that changes can take effect.

This ends the basic configuration that must be done to for the Identity Server to trust the ADFS server as an identity provider. However, the ADFS server needs to be configured to act as an identity server and to trust the Identity Server. Continue with Section 10.2.2, "Configuring the ADFS Server to Be an Identity Provider," on page 270.

10.2.2 Configuring the ADFS Server to Be an Identity Provider

The following tasks describe the minimum configuration required for the ADFS server to act as an identity provider for the Access Manager Identity Server.

- "Enabling a Claim Type for a Resource Partner" on page 270
- "Creating a Resource Partner" on page 270

For additional configuration options, see Section 10.2.4, "Additional WS Federation Configuration Options," on page 271.

Enabling a Claim Type for a Resource Partner

You can enable three types of claims for identity on an ADFS Federation server. They are Common Name, E-mail, and User Principal Name. The ADFS step-by-step guide specifies that you do everything with a User Principal Name, which is an Active Directory convention. Although it could be given an e-mail that looks the same, it is not. This scenario selects to use E-mail instead of Common Name because E-mail is a more common configuration.

1. In the Administrative Tools, open the *Active Directory Federation Services* tool.
2. Navigate to the *Organizational Claims* by clicking *Federation Service* > *Trust Policy* > *My Organization*.
3. Make sure that E-mail is in this list.
4. Navigate to Active Directory by clicking *Federation Services* > *Trust Policy* > *Account Stores*.
5. Enable the *E-mail Organizational Claim*:
   - **5a** Right-click this claim, then select *Properties*.
   - **5b** Click the *Enabled* box.
   - **5c** Add the LDAP mail attribute by clicking *Settings* > *LDAP attribute* and selecting *mail*.

     This is the LDAP attribute in Active Directory where the user's e-mail address is stored.
   - **5d** Click *OK*.
6. Verify that the user you are going to use for authentication has an E-mail address in the mail attribute.
7. Continue with "Creating a Resource Partner" on page 270.

Creating a Resource Partner

The WS Federation protocol requires a two-way trust. The identity provider must be configured to trust the service provider, and the service provider must be configured to trust the identity provider. You have already set up the service provider to trust the identity provider (see "Creating a WS Federation Identity Provider" on page 267). This section sets up the trust so that the identity provider (the ADFS server) trusts the service provider (the Identity Server).

1. In the Active Directory Federation Services console, access the Resource Partners page by clicking *Federation Services* > *Trust Policy* > *Partner Organizations*.
2. Right-click the *Partner Organizations*, then click *New* > *Resource Partner*.
3. Supply the following information in the wizard:
   - You do not have a resource partner policy file to import.

- For the display name, specify the DNS name of the Identity Server.
- For the *Federation Services URI*, enter the following:

  `https://<DNS_Name>:8443/nidp/wsfed/`

  Replace <*DNS_Name*> with the name of your Identity Server.

  This is the base URL of your Identity Server with the addition of `/wsfed/` at the end.
- For the Federation Services endpoint URL, specify the following:

  `https://<DNS_Name>:8443/nidp/wsfed/spassertion_consumer`

  Replace <*DNS_Name*> with the name of your Identity Server.

  This is the base URL of your Identity Server with the addition of `/wsfed/spassertion_consumer` at the end.
- Select *Federated Web SSO*.

  The Identity Server is outside of any forest, so do not select *Forest Trust*.
- Select the E-mail claim.
- Select the *Pass all E-mail suffixes through unchanged* option.

4 Enable this resource partner.

5 Finish the wizard.

6 To test the configuration, continue with Section 10.2.3, “Logging In,” on page 271.

10.2.3 Logging In

1 In a client browser, enter the base URL of your Identity Server.

2 From the list of cards, select the Adatum contract.

3 (Conditional) If you are not joined to the Adatum domain, enter a username and password in the browser pop-up. Use a name and a password that are valid in the Adatum domain.

If you are using the client that is joined to the Adatum domain, the card uses a Kerberos ticket to authenticate to the ADFS identity provider (resource partner).

4 When you are directed back to the Identity Server for Federation User Identification, log in to the Identity Server with a username and password that is valid for the Identity Server (the service provider).

5 Verify that you are authenticated.

6 Close the browser.

7 Log in again.

This time you are granted access without entering credentials at the service provider.

10.2.4 Additional WS Federation Configuration Options

You can enable the sharing of attribute information from the Identity Server to the ADFS server. This involves creating an attribute set and enabling the sending of the attributes at authentication. See Section 10.4.2, “Configuring the Attributes Obtained at Authentication,” on page 274.

For other options that can be modified after you have created the trusted identity server configuration, see Section 10.4, “Modifying a WS Federation Identity Provider,” on page 273.

10.3 Managing WS Federation Providers

The WS Federation page allows you to create or edit trusted identity providers and trusted service providers. When you create an identity provider configuration, you are configuring the Identity Server to be a WS Federation resource partner. When you create a service provider configuration, you are configuring the Identity Server to be a WS Federation account partner.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation*.
2. Select one of the following actions:

   New: Launches the Create Trusted Identity Provider Wizard or the Create Trusted Service Provider Wizard, depending on your selection. For more information, see one of the following:

   - Section 10.3.1, "Creating an Identity Provider for WS Federation," on page 272
   - Section 10.3.2, "Creating a Service Provider for WS Federation," on page 273

   Delete: Allows you to delete the selected identity or service provider. This action deletes the definition.

   Enable: Enables the selected identity or service provider.

   Disable: Disables the selected identity or service provider. When the provider is disabled, the server does not load the definition. However, the definition is not deleted.

   Modify: Click the name of a provider. For configuration information, see Section 10.4, "Modifying a WS Federation Identity Provider," on page 273 or Section 10.5, "Modifying a WS Federation Service Provider," on page 277.
3. Click *OK*, then update the Identity Server.

10.3.1 Creating an Identity Provider for WS Federation

In order to have a trust relationship, you need to set up the ADFS server as an identity provider for the Identity Server.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation*.
2. On the WS Federation page, click *New*, select *Identity Provider*, then fill in the following fields:

   Name: Specify a name that identifies the identity provider, such as `Adatum`.

   Provider ID: Specify the federation service URI of the identity provider, for example `urn:federation:adatum`.

   Sign-on URL: Specify the URL for logging in, such as `https://adfsaccount.adatum.com/adfs/ls/`.

   Logout URL: Specify the URL for logging out, such as `https://adfsresource.treyresearch.net/adfs/ls/`

   Identity Provider: Specify the path to the signing certificate of the ADFS server.
3. Confirm the certificate, then click *Next*.
4. For the authentication card, specify the following values:

   ID: Leave this field blank.

   Text: Specify a description that is available to the user when the user mouses over the card.

   Image: Select an image, such as *Customizable*, or any other image.

Show Card: Enable this option so that the card can be presented to the user as a login option.

5 Click *Finish*.

For information about additional configuration steps required to use this identity provider, see Section 10.2, “Using the ADFS Server as an Identity Provider for an Access Manager Protected Resource,” on page 266.

10.3.2 Creating a Service Provider for WS Federation

In order to establish a trusted relationship with the ADFS server, you need to set up the ADFS server as service provider. The trusted relationship allows the service provider to trust the Identity Server for user authentication credentials.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation*.

2 Click *New* > *Service Provider*, then fill in the following fields:

Name: Specify a name that identifies the service provider, such as `TreyResearch`.

Provider ID: Specify the provider ID of the ADFS server. The default value is `urn:federation:treyresearch`.

Sign-on URL: Specify the URL that the user is redirected to after login. The default value is `https://adfsresource.treyresearch.net/adfs/ls/`.

Logout URL: (Optional) Specify the URL that the user can use for logging out. The default value is `https://adfsresource.treyresearch.net/adfs/ls`.

Service Provider: Specify the path to the signing certificate of the ADFS server.

3 Click *Next*, confirm the certificate, then click *Finish*.

For information about additional configuration steps required to use this service provider, see Section 10.1, “Using the Identity Server as an Identity Provider for ADFS,” on page 255.

10.4 Modifying a WS Federation Identity Provider

This section explains how to modify a WS Federation identity provider after it has been created. Section 10.3.1, “Creating an Identity Provider for WS Federation,” on page 272 explains the steps required to create an identity provider. You can modify the following configuration details:

10.4.1 Renaming the Trusted Provider

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Provider Name]*.

2 In the *Name* field, specify a new name for the trusted provider.

3 Click *OK* twice, then update the Identity Server.

10.4.2 Configuring the Attributes Obtained at Authentication

When the Identity Server creates its request to send to the identity provider, it uses the attributes that you have selected. The request asks the identity provider to provide values for these attributes. You can then use these attributes to create policies, to match user accounts, or if you allow provisioning, to create a user account on the service provider.

To select the attributes:

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Identity Provider]* > *Attributes.*

2 (Conditional) To create an attribute set, select *New Attribute Set* from the *Attribute Set* drop-down menu.

An attribute set is a group of attributes that can be exchanged with the trusted provider. For example, you can specify that the local attribute of any attribute in the Liberty profile (such as Informal Name) matches the remote attribute specified at the service provider.

2a Specify a set name, then click *Next*.

2b On the Define Attributes page, click *New*.

2c Select a local attribute.

2d Specify the name of the remote attribute.

2e For the namespace, specify *http://schemas.xmlsoap.org/claims*.

2f Click *OK*.

2g To add other attributes to the set, repeat Step 2b through Step 2e.

2h Click *Finish*.

3 Select an attribute set.

4 Select attributes from the *Available* list, and move them to the left side of the page.

5 (Conditional) If you created a new attribute set, it must be enabled for STS.

For more information, see "Enabling the Attribute Set" on page 258.

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

10.4.3 Modifying the User Identification Method

The user identification method specifies how to identify the user.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Identity Provider]* > *User Identification.*

2 Select the contract that can be used for authentication. Fill in the following field:

Satisfies contract: Specifies the contract that is satisfied by the assertion received from the identity provider. WS Federation expects the URI name of the contract to look like a URL, so it rejects all default Access Manager contracts. You must create a contract with a URI that conforms to WS Federation requirements.

For more information on how to create this contract, see "Creating a New Authentication Contract" on page 256.

3 Specify whether the user can associate (federate) an account at the identity provider (the ADFS server) with an account at Identity Server. Fill in the following field:

Allow federation: Indicates whether account federation is allowed. Enabling this option assumes that a user account exists at the provider or that a method is provided to create an account that can be associated with the user on subsequent logins. If you do not use this feature, authentication is permitted but is not associated with a particular user account.

4 Select one of the following methods for user identification:

- **Do nothing:** Allows the user to authenticate without creating an association with a user account. This option cannot be used when federation is enabled.
- **Authenticate:** Allows the user to authenticate using a local account.
  - **Allow 'Provisioning':** Provides a button that the user can click to create an account when the authentication credentials do not match an existing account.
- **Provision account:** Allows a new account to be created for the user when the authenticating credentials do not match an existing user. When federation is enabled, the new account is associated with the user and used with subsequent logins. When federation is not enabled, a new account is created every time the user logs in.

  This option requires that you specify a user provisioning method.
- **Attribute matching:** Enables account matching. The service provider can uniquely identify a user in its directory by obtaining specific user attributes sent by the trusted identity provider. This option requires that you specify a user matching method.
  - **Prompt for password on successful match:** Specifies whether to prompt the user for a password when the user's name is matched to an account, to ensure that the account matches.

5 (Conditional) If you selected a method that requires provisioning (*Allow 'Provisioning'* or *Provision account*), click the *Provision settings* icon and create a provisioning method.

For configuration information, see Section 11.3, "Defining the User Provisioning Method," on page 286.

6 (Conditional) If you selected *Attribute matching* as the identification method, click the *Attribute Matching settings* icon and create a matching method.

For configuration information, see Section 11.1.2, "Configuring the Attribute Matching Method for Liberty or SAML 2.0," on page 283.

7 Click *OK* twice, then update the Identity Server.

10.4.4 Viewing the WS Identity Provider Metadata

You can view the metadata of the ADFS server, edit it, and view information about the signing certificate.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Identity Provider]* > *Metadata.*

The following values need to be configured accurately:

ID: This is provider ID. The ADFS server provides this value to the service provider in the realm parameter in the assertion. You set this value in the *Properties* of the *Trust Policy* on the ADFS server. The label is *Federation Service URI*. The default value is `urn:federation:adatum`.

sloUrl: This is the sign-on URL. This URL is listed in the *Properties* of the *Trust Policy* on the ADFS server. The label is *Federation Services endpoint URL*.

ssoUrl: This is the logout URL. The default value is `https://adfsresource.treyresearch.net/adfs/ls/`. The ADFS server makes no distinction between the login URL and the logout URL.

If the values do not match the ADFS values, you need to edit the metadata.

2. To edit the metadata, click *Edit*. For configuration information, see Section 10.4.5, "Editing the WS Identity Provider Metadata," on page 276.
3. To view information about the signing certificate, click *Certificates*.
4. Click *OK* twice.

10.4.5 Editing the WS Identity Provider Metadata

You can view and edit the metadata of the ADFS server.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Identity Provider]* > *Metadata* > *Edit.*
2. Configure the following fields:

   Provider ID: This is the provider ID. The ADFS server provides this value to the service provider in the realm parameter in the assertion. You set this value in the *Properties* of the *Trust Policy* on the ADFS server. The label is *Federation Service URI*. The default value is `urn:federation:adatum`.

   Sign-on URL: This is the sloUrl. This URL is listed in the *Properties* of the *Trust Policy* on the ADFS server. The label is *Federation Services endpoint URL*.

   Logout URL: This is the ssoUrl. The default value is `https://adfsresource.treyresearch.net/adfs/ls/`. The ADFS server makes no distinction between the login URL and the logout URL.
3. If you need to import a new signing certificate, click the *Browse* button and follow the prompts.
4. To view information about the signing certificate, click *Certificates*.
5. Click *OK* twice, then update the Identity Server.

10.4.6 Modifying the Authentication Card

When you create an identity provider, you must also configure an authentication card. After it is created, you can modify it.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Identity Provider]* > *Authentication Card.*
2. Modify the values in one or more of the following fields:

ID: If you have need to reference this card outside of the Administration Console, specify an alphanumeric value here. If you do not assign a value, the Identity Server creates one for its internal use. The internal value is not persistent. Whenever the Identity Server is rebooted, the value can change. A specified value is persistent.

Text: Specify the text that is displayed on the card. This value, in combination with the image, indicates to the users the provider they are logging into.

Image: Specify the image to be displayed on the card. Select the image from the drop-down list. To add an image to the list, click <*Select local image*>.

Show Card: Determine whether the card is shown to the user, which allows the user to select and use the card for authentication. If this option is not selected, the card is only used when a service provider makes a request for the card.

Passive Authentication Only: Select this option if you do not want the Identity Server to prompt the user for credentials. If the user has already authenticated and the credentials satisfy the requirements of this contract, the user is passively authenticated. If the user's credentials do not satisfy the requirements of this contract, the user is denied access.

3 Click *OK* twice, then update the Identity Server.

10.5 Modifying a WS Federation Service Provider

This section explains how to modify a WS Federation service provider after it has been created. Section 10.3.2, "Creating a Service Provider for WS Federation," on page 273 explains the steps required to create the service provider. You can modify the following configuration details:

- Section 10.5.1, "Renaming the Service Provider," on page 277
- Section 10.5.2, "Configuring the Attributes Sent with Authentication," on page 277
- Section 10.5.3, "Modifying the Authentication Response," on page 278
- Section 10.5.4, "Viewing the WS Service Provider Metadata," on page 279
- Section 10.5.5, "Editing the WS Service Provider Metadata," on page 279

10.5.1 Renaming the Service Provider

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Service Provider]*.

2 In the *Name* field, specify a new name for the service provider.

3 Click *OK* twice, then update the Identity Server.

10.5.2 Configuring the Attributes Sent with Authentication

When the Identity Server creates its response for the service provider, it uses the attributes listed on the Attributes page. The response needs to contain the attributes that the service provider requires. If you do not own the service provider, you need to contact the administrator of the service provider and negotiate which attributes you need to send in the response. The service provider can then use these attributes to identify the user, to create policies, to match user accounts, or if it allows provisioning, to create a user account on the service provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Service Provider]* > *Attributes*.

2 (Conditional) To create an attribute set, select *New Attribute Set* from the *Attribute Set* drop-down menu.

An attribute set is a group of attributes that can be exchanged with the trusted provider. For example, you can specify that the local attribute of any attribute in the Liberty profile (such as Informal Name) matches the remote attribute specified at the service provider.

2a Specify a set name, then click *Next*.

2b On the Define Attributes page, click *New*.

2c Select a local attribute.

2d Specify the name of the remote attribute.

2e For the namespace, specify `http://schemas.xmlsoap.org/claims`.

2f Click *OK*.

2g To add other attributes to the set, repeat Step 2b through Step 2e.

2h Click *Finish*.

3 Select an attribute set.

4 Select attributes that you want to send from the *Available* list, and move them to the left side of the page.

5 (Conditional) If you created a new attribute set, it must be enabled for STS.

For more information, see "Enabling the Attribute Set" on page 258.

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

10.5.3 Modifying the Authentication Response

When the Identity Server sends its response to the service provider, the response can contain an identifier for the user. If you do not own the service provider, you need to contact the administrator of the service provider and negotiate whether the user needs to be identified and how to do the identification. If the service provider is going to use an attribute for user identification, that attribute needs to be in the attributes sent with authentication. See Section 10.5.2, "Configuring the Attributes Sent with Authentication," on page 277.

To select the user identification method to send in the response:

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Service Provider]* > *Authentication Response.*

2 For the format, select one of the following:

Unspecified: Specifies that the SAML assertion contains an unspecified name identifier.

E-mail: Specifies that the SAML assertion contains the user's e-mail address for the name identifier.

X509: Specifies that the SAML assertion contains an X.509 certificate for the name identifier.

3 For the value, select an attribute that matches the format. For the Unspecified format, select the attribute that the service provider expects.

The only values available are from the attribute set that you have created for WS Federation.

4 To specify that this Identity Server must authenticate the user, disable the *Use proxied requests* option. When the option is disabled and the Identity Server cannot authenticate the user, the user is denied access.

When this option is enabled, the Identity Server checks to see if other identity providers can satisfy the request. If one or more can, the user is allowed to select which identity provider performs the authentication. If a proxied identity provider performs the authentication, it sends the response to the Identity Server. The Identity Server then sends the response to the service provider.

5 Click *OK* twice, then update the Identity Server.

10.5.4 Viewing the WS Service Provider Metadata

You can view the metadata of the ADFS server, edit it, and view information about the signing certificate.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Service Provider]* > *Metadata.*

The following values need to be configured accurately:

ID: This is provider ID. This is the value that the ADFS server provides to the Identity Server in the realm parameter of the query string. This value is specified in the *Properties* of the *Trust Policy* page on the ADFS server. The parameter label is *Federation Service URI*. The default value is `urn:federation:treyresearch`.

sloUrl: This is the sign-on URL. This URL is listed in the *Properties* of the *Trust Policy* on the ADFS server. The label is *Federation Services endpoint URL*. The default value is `https://adfsresource.treyresearch.net/adfs/ls/`.

ssoUrl: This is the logout URL. The default value is `https://adfsresource.treyresearch.net/adfs/ls/`. The ADFS server makes no distinction between the login URL and the logout URL.

If the values do not match the ADFS values, you need to edit the metadata.

2 To edit the metadata, click *Edit*. For configuration information, see Section 10.5.5, “Editing the WS Service Provider Metadata,” on page 279.

3 To view information about the signing certificate, click *Certificates*.

4 Click *OK* twice.

10.5.5 Editing the WS Service Provider Metadata

You can view the metadata of the ADFS server and edit metadata.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *WS Federation* > *[Service Provider]* > *Metadata* > *Edit.*

2 Configure the following fields:

Provider ID: This is provider ID. This is the value that the ADFS server provides to the Identity Server in the realm parameter of the query string. This value is specified in the *Properties* of the *Trust Policy* page on the ADFS server. The parameter label is *Federation Service URI*. The default value is `urn:federation:treyresearch`.

Sign-on URL: This is the sloUrl. This URL is listed in the *Properties* of the *Trust Policy* on the ADFS server. The label is *Federation Services endpoint URL*. The default value is `https://adfsresource.treyresearch.net/adfs/ls/`.

Logout URL: This is the ssoUrl. The default value is `https://adfsresource.treyresearch.net/adfs/ls/`. The ADFS server makes no distinction between the login URL and the logout URL.

3. If you need to import a new signing certificate, click the Browse button and follow the prompts.
4. To view information about the signing certificate, click *Certificates*.
5. Click *OK* twice, then update the Identity Server.

Configuring User Identification Methods for Federation

11

Configuring authentication involves determining how the service provider interacts with the identity provider during user authentication and federation. Three methods exist for you to identify users from a trusted identity provider:

- You can identify users by matching their authentication credentials
- You can match selected attributes and then prompt for a password to verify the match, or you can use just the attributes for the match.
- You can assume that the user does not have an account and create new accounts with user provisioning. You can also allow for provisioning when the matching methods fail. If there are problems during provisioning, you see error messages with more information.

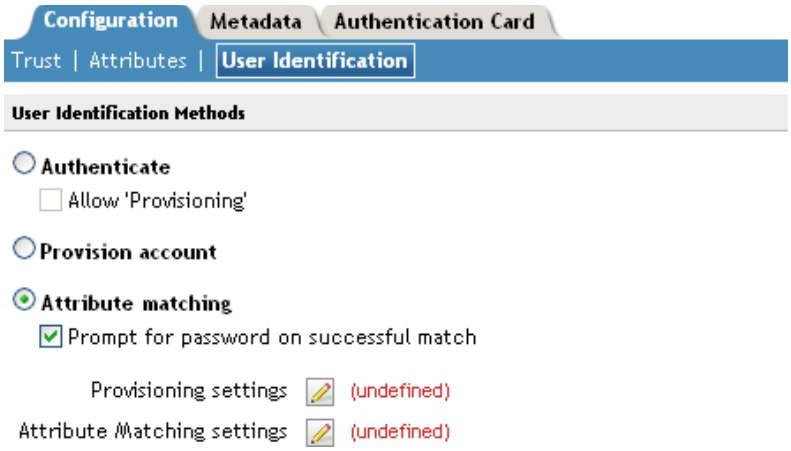
The following sections describe how to configure these methods:

11.1 Defining User Identification for Liberty and SAML 2.0

11.1.1 Selecting a User Identification Method for Liberty or SAML 2.0

User identification determines how an account at the identity provider is matched with an account at the service provider. If federation is enabled between the two, the user can set up a permanent relationship between the two accounts. If federation is not enabled (see Section 7.8, "Configuring an Authentication Request for an Identity Provider," on page 211), you cannot set up a user identification method.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty [or SAML 2.0]* > *[Identity Provider]* > *User Identification*.



2 Specify how users are identified on the SAML 2.0 or Liberty provider. Select one of the following methods:

- **Authenticate:** Select this option when you want to use login credentials. This option prompts the user to log in at both the identity provider and the service provider on first access. If the user selects to federate, the user is prompted, on subsequent logins, to authenticate only to the identity provider.
  - **Allow 'Provisioning':** Select this option to allow users to create an account when they have no account on the service provider.

    This option requires that you specify a user provisioning method.
- **Provision Account:** Select this option when the users on the identity provider do not have accounts on the service provider. This option allows the service provider to trust any user that has authenticated to the trusted identity provider

  This option requires that you specify a user provisioning method.
- **Attribute matching:** Select this option when you want to use attributes to match an identity server account with a service provider account. This option requires that you specify a user matching method.
  - **Prompt for password on successful match:** Select this option to prompt the user for a password when the user's name is matched to an account, to ensure that the account matches.

3 Select one of the following:

- If you selected the *Attribute matching* option, select a method, then click *OK*. If you have not created a matching method, continue with Section 11.1.2, "Configuring the Attribute Matching Method for Liberty or SAML 2.0," on page 283.
- If you selected the *Provision account* option, select a method, then click *OK*. If you have not created a provisioning method, continue with Section 11.3, "Defining the User Provisioning Method," on page 286.
- If you selected the *Authenticate* option with the *Allow Provisioning* option, select a method, then click *OK*. If you have not created a provisioning method, continue with Section 11.3, "Defining the User Provisioning Method," on page 286.
- If you selected the *Authenticate* option without the *Allow Provisioning* option, click *OK*.

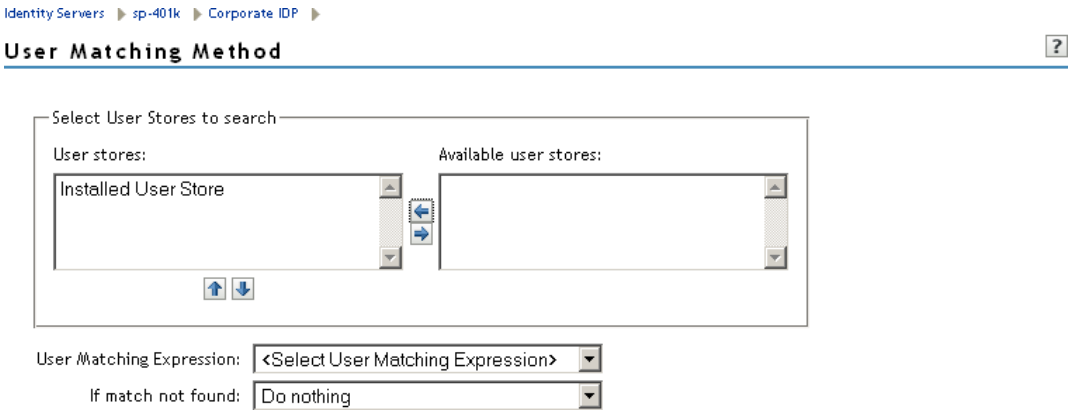
4 Click *OK*, then update the Identity Server.

11.1.2 Configuring the Attribute Matching Method for Liberty or SAML 2.0

If you enabled the *Attribute matching* option when selecting a user identification method, you must configure a matching method.

The Liberty Personal Profile is enabled by default. If you have disabled it, you need to enable it. See Section 13.2, "Managing Web Services and Profiles," on page 296.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Servers* > *Edit* > *Liberty [or or SAML 2.0]* > *[Identity Provider]* > *User Identification*.
2. Click *Attribute Matching settings*.



3. Select and arrange the user stores you want to use.

   Order is important. The user store at the top of the list is searched first. If a match is found, the other user stores are not searched.
4. Select a matching expression, or click *New* to create a look-up expression. For information on creating a look-up expression, see Section 6.3, "Configuring User Matching Expressions," on page 180.
5. Specify what action to take if no match is found.
   - **Do nothing:** Specifies that an identity provider account is not matched with a service provider account. This option allows the user to authenticate the session without identifying a user account on the service provider.

     IMPORTANT: Do not select this option if the expected name format identifier is persistent. A persistent name format identifier requires that the user be identified so that information can be stored with that user. To support the *Do nothing* option and allow anonymous access, the authentication response must be configured for a transient identifier format. To view the service provider configuration, see Section 7.9, "Configuring an Authentication Response for a Service Provider," on page 216.
   - **Prompt user for authentication:** Allows the user to specify the credentials for a user that exists on the service provider. Sometimes users have accounts at both the identity provider and the service provider, but the accounts were created independently, use different names (for example, joe.smith and jsmith) and different passwords, and share no common attributes except for the credentials known by the user.

- **Provision account:** Assumes that the user does not have an account at the service provider and creates one for the user. You must create a provisioning method.

6 Click *OK*.

7 (Conditional) If you selected *Provision account* when no match is found, select the *Provision settings* icon. For information on this process, see Section 11.3, "Defining the User Provisioning Method," on page 286.

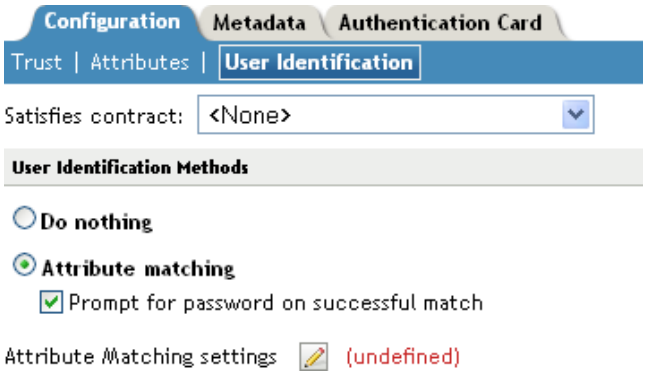
8 Click *OK* twice, then update the Identity Server.

11.2 Defining User Identification for SAML 1.1

11.2.1 Selecting a User Identification Method for SAML 1.1

Two methods exist for identifying users from an identity provider when using the SAML 1.1 protocol. You can specify that no account matching needs to occur, or you can configure a match method. You configure a match method when you want to use attributes from the identity provider to uniquely identify a user on the service provider.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 1.1* > *[Identity Provider]* > *User Identification*.



2 In the *Satisfies contract* option, specify the contract that can be used to satisfy the assertion received from the identity provider. Because SAML 1.1 does not use contracts and because the Identity Server is contract-based, this setting permits an association to be made between a contract and a SAML 1.1 assertion.

Use caution when assigning the contract to associate with the assertion, because it is possible to imply that authentication has occurred, when it has not. For example, if a contract is assigned to the assertion, and the contract has two authentication methods (such as one for name/password and another for X.509), the server sending the assertion might use only name/password, but the service provider might assume that X.509 took place and then incorrectly assert it to another server.

3 Select one of the following options for user identification:

- **Do nothing:** Specifies that an identity provider account is not matched with a service provider account. This option allows the user to authenticate the session without identifying a user account on the service provider.

- **Attribute matching:** Authenticates a user by matching a user account on the identity provider with an account on the service provider. This option requires that you set up the match method.
  - **Prompt for password on successful match:** Specifies whether to prompt the user for a password when the user is matched to an account, to ensure that the account matches.

4 Select one of the following:

- If you selected *Do nothing*, continue with Step 5.
- If you selected *Attribute matching*, continue with Section 11.2.2, "Configuring the Attribute Matching Method for SAML 1.1," on page 285.

5 Click *OK* twice.

6 Update the Identity Server.

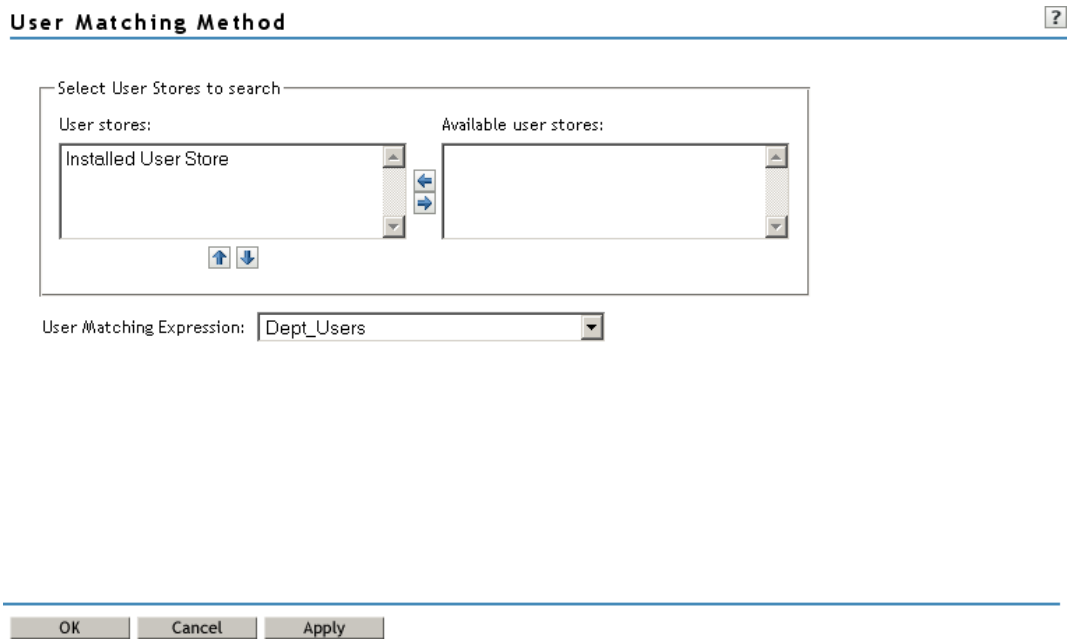
11.2.2 Configuring the Attribute Matching Method for SAML 1.1

A user matching expression is a set of logic groups with attributes that uniquely identify a user. User matching expressions enable you to map the Liberty attributes to the correct LDAP attributes during searches. You must know the LDAP attributes that can be used to identify unique users in the user store.

In order to use user matching, the Personal Profile must be enabled. It is enabled by default. If you have disabled it, you need to enable it. See Section 13.2, "Managing Web Services and Profiles," on page 296.

1 In the Administration Console, click *Devices > Identity Servers > Servers > Edit > SAML 1.1 > [Identity Provider] > User Identification.*

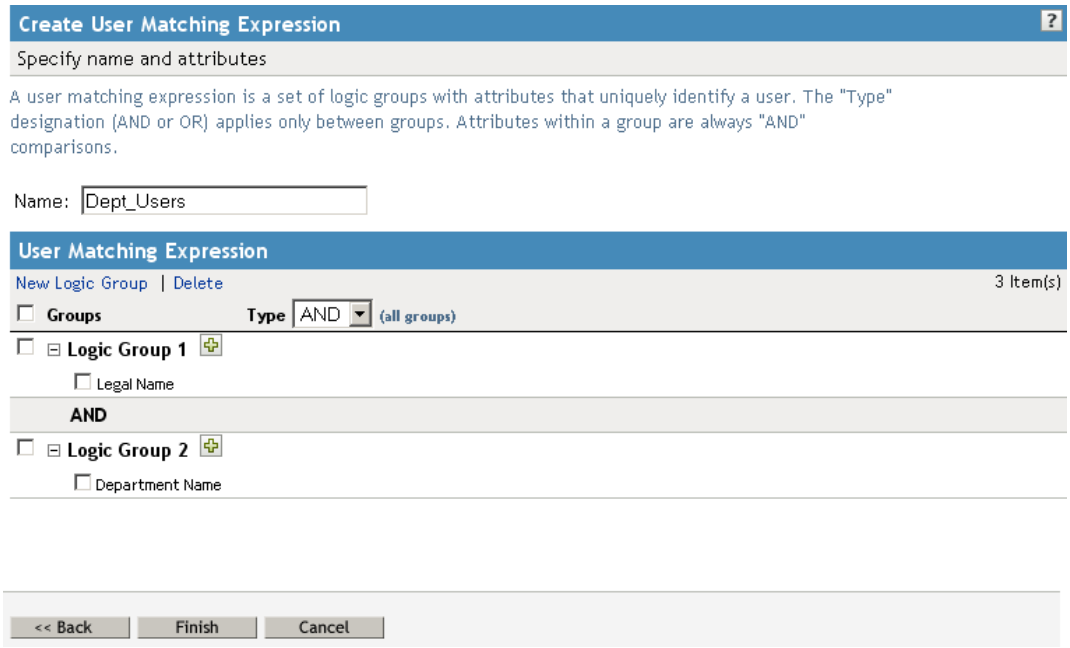
2 To configure the match method, click *Attribute Matching settings*.



3 To configure user matching, fill in the following fields:

Select User Stores to search: Select and order the user stores you want to use in the search.

User Matching Expression: Select a matching expression, or click *New User Matching Expression* to create one.



3a In the *Name* option, specify a name for the matching expression.

3b Click the *Add Attributes* icon, then select an attribute.

The Personal Profile attributes are listed first, then the LDAP attributes.

3c (Conditional) To add more attributes, click the *Add Attributes* icon.

3d Click *Finish*.

3e Select the new expression on the User Method Matching page, then click *OK*.

4 Click *OK* twice.

5 Update the Identity Server.

11.3 Defining the User Provisioning Method

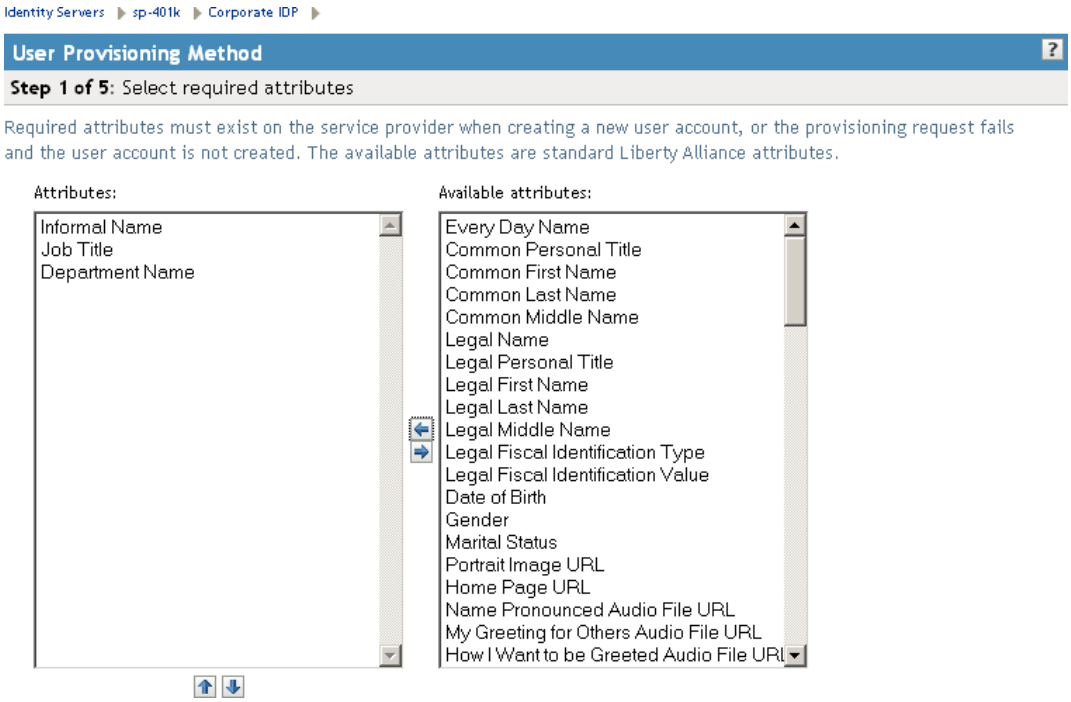
If you have selected *Provision account* as the user identification method or have created an attribute matching setting that allows for provisioning when no match is found, you need to create a provision method. This procedure involves selecting required and optional attributes that the service provider requests from the identity provider during provisioning.

IMPORTANT: When a user object is created in the directory, some attributes are initially created with the value of NAM Generated. Afterwards, an attempt is made to write the required and optional attributes to the new user object. Because required and optional attributes are profile attributes, the system checks the write policy for the profile's Data Location Settings (specified in *Liberty > Web Service Provider*) and writes the attribute in either LDAP or the configuration store. In order for the

LDAP write to succeed, each attribute must be properly mapped as an LDAP Attribute. Additionally, you must enable the read/write permissions for each attribute in the Liberty/LDAP attribute maps. See Section 13.6, “Mapping LDAP and Liberty Attributes,” on page 308.

To configure user provisioning:

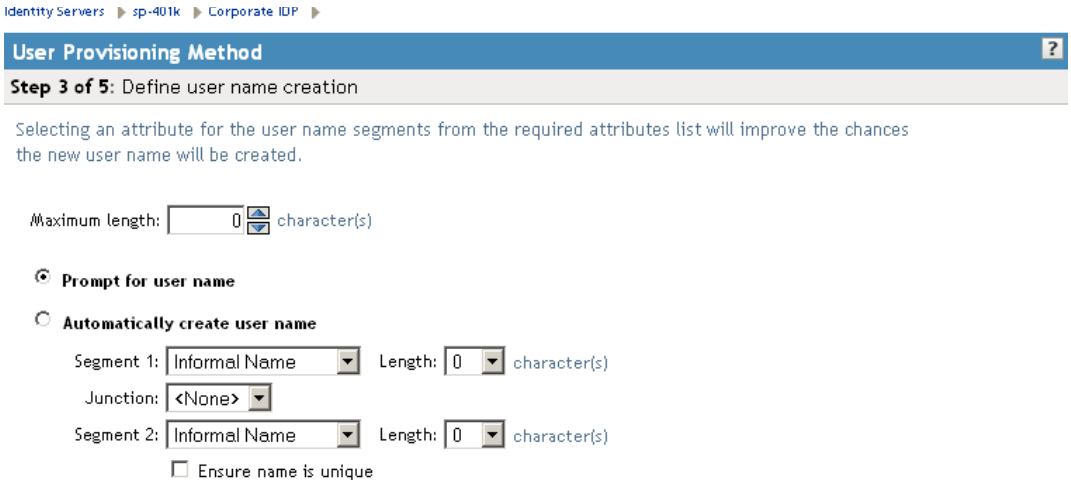
1. In the Administration Console, click *Devices* > *Identity Servers* > *Servers* > *Edit* > *Liberty [or SAML 2.0]* > *[Identity Provider]* > *User Identification*.
2. Click the *Provisioning settings* icon.



3. Select the required attributes from the *Available Attributes* list and move them to the *Attributes* list.

   Required attributes are those used in the creation of a user name, or that are required when creating the account.
4. Click *Next*.
5. Select optional attributes from the *Available Attributes* list and move them to the *Attributes* list.

   This step is similar to selecting required attributes. However, the user provisioning request creates the user account whether or not the optional attributes exist on the service provider.
6. Click *Next*.
7. Define how to create the username.



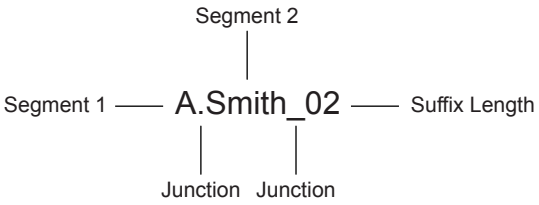
You can specify whether users are prompted to create their own usernames or whether the system automatically creates usernames. Selecting an attribute for the username segments from the required attributes list improves the chances that a new username is successfully created.

Maximum length: The maximum length of the user name. This value must be between 1 and 50.

Prompt for user name: Enables users to create their own usernames.

Automatically create user name: Specifies that the system creates usernames. You can configure the segments for the system to use when creating usernames and configure how the names are displayed.

For example, if you are using the required attributes of Common First Name and Common Last Name, a username for Adam Smith might be generated as A.Smith_02, as shown in the following illustration:



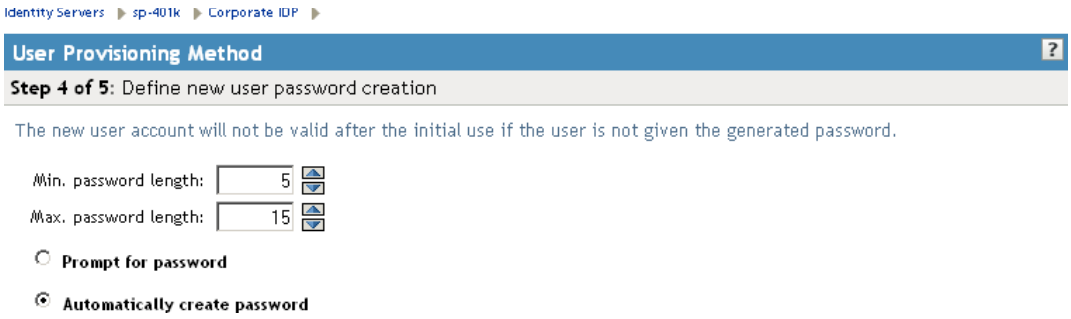
Use the following settings to specify how this is accomplished:

- **Segment 1:** The required attribute to use as the first segment for the user name. The values displayed in this drop-down menu correspond to the required attributes you selected. For example, you might select Common First Name to use for *Segment 1*.
- **Length:** The length of the first attribute segment. For example, if you selected Common First Name for the *Segment 1* value, setting the length to 1 specifies that the system uses the first letter of the Common First Name attribute. Therefore, Adam Smith would be ASmith.
- **Junction:** The type of junction to use between the attributes of the user name. If a period is selected, Adam Smith would display as A.Smith.
- **Segment 2:** The required attribute to use as the second segment for the user name. The values displayed in this drop-down menu correspond to the required attributes you selected. For example, you might select Common Last Name to use for *Segment 2*.

- **Length:** The length of the second attribute segment. For example, if you selected Common Last Name for the *Segment 2* value, you might set the length to *All*, so that the full last name is displayed. However, the system does not allow more than 20 characters for the length of segment 2.
- **Ensure name is unique:** Applies a suffix to the colliding name until a unique name is found, if using attributes causes a collision with an existing name. If no attributes are provided, or the lengths for them are 0, and this option is selected, the system creates a unique name.

8 Click *Next*.

9 Specify password settings.



Use this page to specify whether to prompt the user for a password or to create a password automatically.

Min. password length: The minimum length of the password.

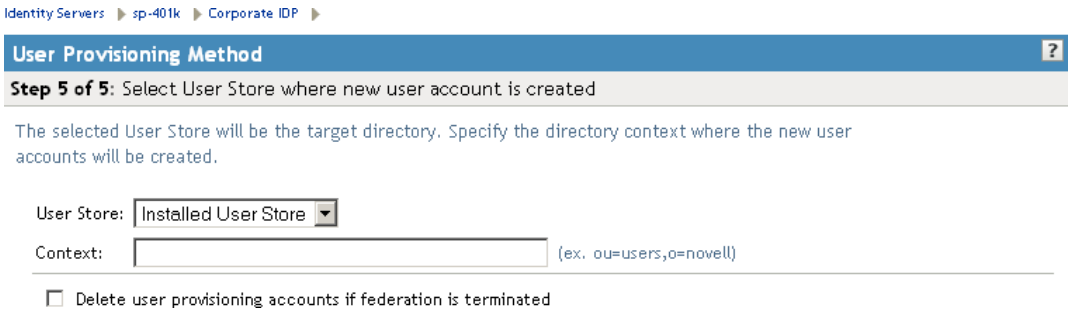
Max. password length: The maximum length of the password.

Prompt for password: Prompts the user for a password.

Automatically create password: Specifies whether to automatically create passwords.

10 Click *Next*.

11 Specify the user store and context in which to create the account.



User Store: The user store in which to create the new user account.

Context: The context in the user store you want accounts created.

The system creates the user within a specific context; however, uniqueness is not guaranteed across the directory.

Delete user provisioning accounts if federation is terminated: Specifies whether to automatically delete the provisioned user account at the service provider if the user terminates his or her federation between the identity provider and service provider.

12 Click *Finish*.

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

11.4 User Provisioning Error Messages

The following error messages are displayed for the end user if there are problems during provisioning.

Table 11-1 *Provisioning Error Messages*

Error Message	Cause
`Username length cannot exceed (?) characters.`	The user entered more characters for a user name than is allowed, as specified by the administrator.
`Username is not available.`	The user entered a name that already exists in the directory.
`Passwords don't match.`	The user provided two password values that do not match.
`Passwords must be between (x) and (y) characters in length.`	The user provided password values that are either too short or too long.
`Username unavailable.`	The provisioned user account was deleted without first defederating the user. Remove orphaned identity objects from the configuration datastore. **IMPORTANT:** Only experienced LDAP users should remove orphaned identity objects from the configuration datastore. You must ensure that the objects you are removing are orphaned. Otherwise, you create orphaned objects by mistake.
`Unable to complete authentication request.`	Can occur when users are allowed to create accounts from a service provider's login page, when the service provider uses Active Directory for the user store. The password provided does not conform to the Windows password complexity policy in Active Directory. Ensure that Active Directory is configured to use a secure port, such as 636, and that the user's password conforms to the complexity policy. If you encounter this error, you must reset the password on the Windows machine.

Configuring Communication Profiles

12

You can configure the methods of communication that are available at the server for requests and responses sent between providers. These settings affect the metadata for the server and should be determined prior to publishing to other sites.

12.1 Configuring a Liberty Profile

The profile specifies what methods of communication are available at the server for the Liberty protocol. These settings affect the metadata for the server and should be determined prior to publishing to other sites. If you have set up trusted providers, and then modify these profiles, the trusted providers need to reimport the metadata from this Identity Server.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Profiles*.
2. Configure the following fields for identity providers and service providers:

   Login: Specifies whether to support Artifact or Post binding for login. Select one or more of the following for the identity provider and the service provider:

   - The *Artifact* binding provides an increased level of security by using a back channel means of communication between the two servers during authentication.
   - The *Post* method uses HTTP redirection to accomplish communication between the servers.

   Single Logout: Specifies the communication method to use when the user logs out. Typically, you select both of these options, which enables the identity provider or service provider to accept both HTTP and SOAP requests. SOAP is used if both options are selected, or if the service provider has not specified a preference.

   - **HTTP:** Uses HTTP 302 redirects or HTTP GET requests to communicate logout requests from this identity site to the service provider.
   - **SOAP:** Uses SOAP over HTTP messaging to communicate logout requests from this identity provider to the service provider.

   Federation Termination: Specifies the communication channel to use when the user selects to defederate an account. Typically, you select both of these options, which enables the identity provider or service provider to accept both HTTP and SOAP requests. SOAP is the default setting if the service provider has not specified a preference.

   - **HTTP:** Uses HTTP 302 redirects to communicate federation termination requests from this server.
   - **SOAP:** Uses SOAP back channel over HTTP messaging to communicate logout requests from this server

Register Name: Specifies the communication channel to use when the provider supplies a different name to register for the user. Typically, you select both of these options, which enables the identity provider or service provider to accept both HTTP and SOAP requests. SOAP is the default setting if the service provider has not specified a preference.

- **HTTP:** Uses HTTP 302 redirects to communicate federation termination requests from this server.
- **SOAP:** Uses SOAP back channel over HTTP messaging to communicate logout requests from this server.

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

4 (Conditional) If you have set up trusted providers and have modified the profile, these providers need to reimport the metadata from this Identity Server.

12.2 Configuring a SAML 1.1 Profile

Profiles control what methods of communication are available at the server for the SAML 1.1 protocol. These settings affect the metadata for the server and should be determined prior to publishing to other sites. If you have set up trusted providers, and then modify these profiles, the trusted providers need to reimport the metadata from this Identity Server.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 1.1* > *Profiles*.

2 Configure the following fields:

Login: Specifies the communication channel when the user logs in. Select one or more of these methods for the identity provider and the identity consumer:

- The Artifact binding provides an increased level of security by using the back channel for communication between the two servers during authentication.
- The Post method uses HTTP redirection to accomplish communication between servers.

Source ID: Displays the hexadecimal ID generated by the Identity Server for the SAML 1.1 service provider. This is a required value when establishing trust with a service provider.

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

4 (Conditional) If you have set up trusted providers and have modified the profile, these providers need to reimport the metadata from this Identity Server.

12.3 Configuring a SAML 2.0 Profile

Profiles control the methods of communication that are available for SAML 2.0 protocol requests and responses sent between trusted providers. These settings affect the metadata for the server and should be determined prior to publishing to other sites. The identity provider uses the incoming metadata to determine how to respond.

All available profile bindings are enabled by default. SOAP is used when all are enabled (or if the service provider has not specified a preference), followed by HTTP Post, then HTTP Redirect.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *SAML 2.0* > *Profiles*.

2 Configure the following fields for identity providers and identity consumers (service providers):

Artifact Resolution: Specify whether to enable artifact resolution for the identity provider and identity consumer.

The assertion consumer service at the service provider performs a back-channel exchange with the artifact resolution service at the identity provider. Artifacts are small data objects pointing to larger SAML protocol messages. They are designed to be embedded in URLs and conveyed in HTTP messages.

Login: Specifies the communication channel to use when the user logs in. Select one or more of the following:

- **Post:** A browser-based method used when the SAML requester and responder need to communicate using an HTTP user agent. This occurs, for example, when the communicating parties do not share a direct path of communication. You also use this when the responder requires user interaction in order to fulfill the request, such as when the user must authenticate to it.
- **Redirect:** A browser-based method that uses HTTP 302 redirects or HTTP GET requests to communicate requests from this identity site to the service provider. SAML messages are transmitted within URL parameters.

Single Logout: Specifies the communication channel to use when the user logs out. Select one or more of the following:

- **HTTP Post:** A browser-based method used when the SAML requester and responder need to communicate by using an HTTP user agent. This occurs, for example, when the communicating parties do not share a direct path of communication. You also use this when the responder requires user interaction in order to fulfill the request, such as when the user must authenticate to it.
- **HTTP Redirect:** A browser-based method that uses HTTP 302 redirects or HTTP GET requests to communicate requests from this identity site to the service provider. SAML messages are transmitted within URL parameters.
- **SOAP:** Uses SOAP back channel over HTTP messaging to communicate requests from this identity provider to the service provider.

Name Management: Specifies the communication channel for sharing the common identifiers for a user between identity and service providers. When an identity provider has exchanged a persistent identifier for the user with a service provider, the providers share the common identifier for a length of time. When either the identity or service provider changes the format or value to identify the user, the system can ensure that the new format or value is properly transmitted. Select one or more of the following:

- **HTTP Post:** A browser-based method used when the SAML requester and responder need to communicate using an HTTP user agent. This occurs, for example, when the communicating parties do not share a direct path of communication. You also use this when the responder requires user interaction in order to fulfill the request, such as when the user must authenticate to it.
- **HTTP Redirect:** A browser-based method that uses HTTP 302 redirects or HTTP GET requests to communicate requests from this identity site to the service provider. SAML messages are transmitted within URL parameters.
- **SOAP:** Uses SOAP back channel over HTTP messaging to communicate requests from this identity provider to the service provider.

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

4 (Conditional) If you have set up trusted providers and have modified these profiles, the providers need to reimport the metadata from this Identity Server.

Configuring Liberty Web Services 13

A Web service uses Internet protocols to provide a service. It is an XML-based protocol transported over SOAP, or a service whose instances and data objects are addressable via URIs.

Access Manager consists of several elements that comprise Web services:

- **Web Service Framework:** Manages all Web services. The framework defines SOAP header blocks and processing rules that enable identity services to be invoked via SOAP requests and responses.
- **Web Service Provider:** An entity that provides data via a Web service. In Access Manager, Web service providers host Web service profiles, such as the Employee Profile, Credential Profile, Personal Profile, and so on.
- **Web Service Consumer:** An entity that uses a Web service to access data. Web service consumers discover resources at the Web service provider, and then retrieve or update information about a user, or on behalf of a user. Resource discovery among trusted partners is necessary because a user might have many kinds of identities (employee, spouse, parent, member of a group), as well as several identity providers (employers or other commercial Web sites).
- **Discovery Service:** The service assigned to an identity provider that enables a Web Service Consumer to determine which Web service provider provides the required resource.
- **LDAP Attribute Mapping:** Access Manager's solution for mapping Liberty attributes with established LDAP attributes.

This section describes the following topics:

For additional resources about the Liberty Alliance specifications, visit the Liberty Alliance Specification (http://www.projectliberty.org/resources/specifications.php) page.

13.1 Configuring the Web Services Framework

The Web Services Framework page lets you edit and manage all the details that pertain to all Web services. This includes the framework for building interoperable identity services, permission-based attribute sharing, identity service description and discovery, and the associated security mechanisms.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Framework*.
2. Fill in the following fields:

   Enable Framework: Enables Web Services Framework.

Axis SOAP Engine Settings: Axis is the SOAP engine that handles all Web service requests and responses. Web services are deployed using XML-based files known as Web service deployment descriptors (WSDD). On startup, Access Manager automatically creates the server-side and client-side configuration for Axis to handle all enabled Web services.

If you need to override this default configuration, use the *Axis Server Configuration WSDD XML* field and the *Axis Client Configuration WSDD XML* field to enter valid WSDD XML. If either or both of these controls contain valid XML, then Access Manager does not automatically create the configuration (server or client) on startup.

3 Click *OK*.

13.2 Managing Web Services and Profiles

After a service has been discovered and authorization data has been received from a trusted identity provider, the Web service consumer can invoke the service at the Web service provider. A Web service provider is the hosting or relying entity on the server side that can make access control decisions based on this authorization data and upon its business practices and preferences.

1 In the Administration Console click *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.

2 Select one of the following actions

New: To create a new Web service, click *New*. This activates the Create Web Service Wizard. You can create a new profile only if you have deleted one.

Delete: To delete an existing profile, select the profile, then click *Delete*.

Enable: To enable a profile, select the profile, then click *Enable*.

Disable: To disable a profile, select the profile, then click *Disable*.

Edit a Policy: To edit the policy associated with a profile, click the *Policy* link. For configuration information, see Section 13.2.4, “Editing Web Service Policies,” on page 301.

Edit a profile: To edit a profile, click the name of a profile. For information on configuring the details, see Section 13.2.1, “Modifying Service and Profile Details for Employee, Custom, and Personal Profiles,” on page 297 and Section 13.2.2, “Modifying Details for Authentication, Discovery, LDAP, and User Interaction Profiles,” on page 299.

For information on modifying the description, see Section 13.2.3, “Editing Web Service Descriptions,” on page 300.

The Identity Server comes with the following Web service profile types:

Authentication Profile: Allows the system to access the roles and authentication contracts in use by current authentications. This profile is enabled by default so that Embedded Service Providers can evaluate roles in policies. This profile 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.

WARNING: Do not delete this profile. In normal circumstances, this profile is used only by the system.

Credential Profile: Allows users to define information to keep secret. It uses encryption to store the data in the directory the user profile resides in.

Custom Profile: Used to create custom attributes for general use.

Discovery: Allows requesters to discover where the resources they need are located. Entities can place resource offerings in a discovery resource, allowing other entities to discover them. Resources might be a personal profile, a calendar, travel preferences, and so on.

Employee Profile: Allows you to manage employment-related information and how the information is shared with others. A company address book that provides names, phones, office locations, and so on, is an example of an employee profile.

LDAP Profile: Allows you to use LDAP attributes for authorization and general use.

Personal Profile: Allows you to manage personal information and to determine how to share that information with others. A shopping portal that manages the user's account number is an example of a personal profile.

User Interaction: Allows you to set up a trusted user interaction service, used for identity services that must interact with the resource owner to get information or permission to share data with another Web service consumer. This profile enables a Web service consumer and Web service provider to cooperate in redirecting the resource owner to the Web service provider and back to the Web service consumer.

3 Click *OK*.

4 On the Servers page, update the Identity Server.

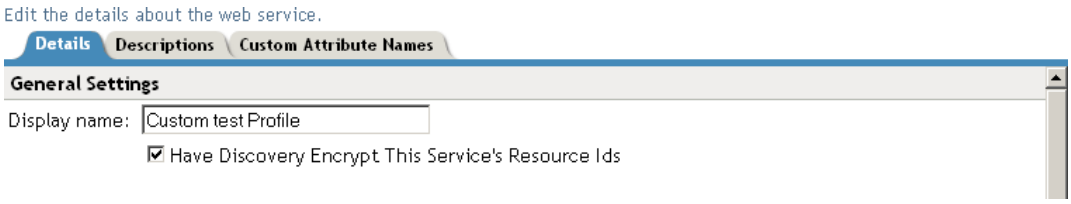
13.2.1 Modifying Service and Profile Details for Employee, Custom, and Personal Profiles

The settings on the Details page are identical for the Employee, Custom, and Personal Profiles. This page allows you to specify the display name, resource ID encryption, and how the system reads and writes data.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.

2 Click *Custom Profile*, *Employee Profile*, or *Personal Profile*, depending on which profile you want to edit.

3 Click the *Details* tab (it is displayed by default).

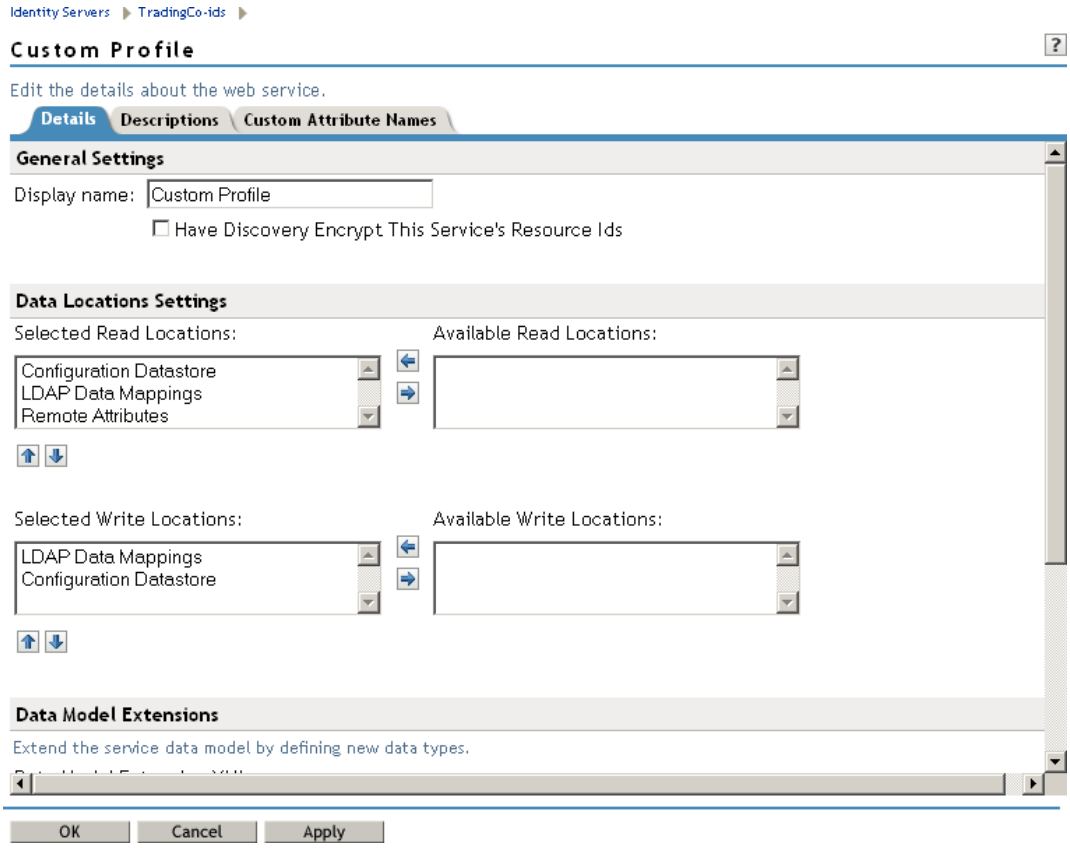


4 Specify the general settings, as necessary:

Display Name: The Web service name. This specifies how the profile is displayed in the Administration Console.

Have Discovery Encrypt This Service's Resource Ids: Specifies whether the Discovery Service encrypts resource IDs. A resource ID is an identifier used by Web services to identify a user. The Discovery Service returns a list of resource IDs when a trusted service provider queries for the services owned by a given user. The Discovery Service has the option of encrypting the resource ID or sending it unencrypted.

5 Specify data location settings:



Selected Read Locations: The list of selected locations from which the system reads attributes containing profile data. If you add multiple entries to this list, the system searches attributes in each location in the order you specify. When a match is found for an attribute, the other locations are not searched. Use the up/down and left/right arrows to control which locations are selected and the order in which to read them. Read locations can include:

- **Configuration Datastore:** Liberty attribute values can be stored in the configuration store of the Administration Console. If your users have access to the User Portal, they can add values to a number of Liberty attributes.
- **LDAP Data Mappings:** If you have mapped a Liberty attribute to an LDAP attribute in your user store, the values can be read from the LDAP user store. To create LDAP attribute maps, see Section 13.6, “Mapping LDAP and Liberty Attributes,” on page 308.
- **Remote Attributes:** If you set up federation, the Identity Server can read attributes from these remote service providers. Sometimes, the service provider is set up to push a set of attribute values when the user logs in. These pushed attributes are cached, and the Identity Server can quickly read them. If a requested attribute has not been pushed, a request for the Liberty attribute is sent to remote service provider. This can be time consuming, especially if the user has federated with more than one remote service provider. *Remote Attributes* should always be the last item in this list.

Available Read Locations: The list of available locations from which the system can read attributes containing profile data. Locations in this list are currently not being used.

Selected Write Locations: The list of selected locations to write attribute data to. If you add multiple entries to this list, the system searches attributes in each location in the order you specify. When a match is found for an attribute, the other locations are not searched. Use the up/down and left/right arrows to control which locations are selected and the order in which they are selected.

- **Configuration Datastore:** Liberty attribute values can be stored in the configuration store of the Administration Console. The Identity Server can write values to these attributes. If this location appears first in the list of *Selected Write Locations*, all Liberty attribute values are written to this location. If you want values written to the LDAP user store, the *LDAP Data Mappings* location must appear first in the list.
- **LDAP Data Mappings:** If you have mapped a Liberty attribute to an LDAP attribute in your user store, the Identity Server can write values to the attribute in the LDAP user store. To create LDAP attribute maps, see Section 13.6, "Mapping LDAP and Liberty Attributes," on page 308.

Available Write Locations: The list of available locations to write attributes containing profile data. Locations in this list are currently not being used.

6 (Optional) Specify data model extensions.

Data Model Extension XML: The data model for some Web services is extensible. You can enter XML definitions of data model extensions in this field. Data model extensions hook into the existing Web service data model at predefined locations.

All schema model extensions reside inside of a schema model extension group. The group exists to bind model data items together under a single localized group name and description. Schema model extension groups can reside inside of a schema model extension root or inside of a schema model extension. There can only be one group per root or extension. Each root is hooked into the existing Web service data model. Multiple roots can be hooked into the same location in the existing Web service data model. This conceptual model applies to the structure of the XML that is required to define data model extensions.

See Appendix C, "Data Model Extension XML," on page 375 for more information.

7 Click *OK*, then click *OK* on the Web Service Provider page.

8 Update the Identity Server.

13.2.2 Modifying Details for Authentication, Discovery, LDAP, and User Interaction Profiles

This page allows you to specify information for Discovery, LDAP, and User Interaction Web service profiles. If you are creating a Web service type, this is Step 2 of the Create Web Service Wizard.

For conceptual information about profiles, see Managing Web Services and Profiles.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider* > *[Profile]*.

2 Click *Authentication, Discovery, LDAP,* or *User Interaction*, depending on which profile you want to edit.

3 Configure the following fields:

Display name: The Web service name. This specifies how the profile is displayed in the Administration Console.

Have Discovery Encrypt This Service's Resource Ids: (Not applicable for the Discovery profile) Specifies whether the Discovery Service encrypts resource IDs. A resource ID is an identifier used by Web services to identify a user. The Discovery Service returns a list of resource IDs when a trusted service provider queries for the services owned by a given user. The Discovery Service has the option of encrypting the resource ID or sending it unencrypted. This ID is encrypted with the public key of the resource provider generated at installation. Encrypting resource IDs is turned off by default.

4 Click *OK*.

13.2.3 Editing Web Service Descriptions

All of the Description pages on each profile are identical. You can define how a service provider gains access to portions of the user's identity information that can be distributed across multiple providers. The service provider uses the Discovery Service to ascertain the location of a specific identity service for a user. The Discovery Service enables various entities to dynamically and securely discover a user's identity service, and it responds, on a permission basis, with a service description of the desired identity service.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.

2 Click the profile or service.

3 Click *Descriptions*.

4 Click the description name, or click *New*.

5 Fill in the following fields:

Name: The Web Service Description name.

Security Mechanism: (Required) Liberty uses channel security (TLS 1.0) and message security in conjunction with the security mechanism. Channel security addresses how communication between identity providers, service providers, and user agents is protected. For authentication, service providers are required to authenticate identity providers by using identity provider server-side certificates. Identity providers have the option to require authentication of service providers by using service provider client-side certificates.

Message security addresses security mechanisms applied to the discrete Liberty protocol messages passed between identity providers, service providers, and user agents.

Select the mechanism for message security. Message authentication mechanisms indicate which profile is used to ensure the authenticity of a message.

- **X.509:** Used for message exchanges that generally rely upon message authentication as the principle factor in making authorization decisions.
- **SAML:** Used for message exchanges that generally rely upon message authentication as well as the conveyance and attestation of authorization information.
- **Bearer:** Based on the presence of the security header of a message. In this case, the bearer token is verified for authenticity rather than proving the authenticity of the message.

6 Under *Select Service Access Method*, select either *Brief Service Access Method* or *WSDL Service Access Method*.

Brief Service Access Method: Provides the information necessary to invoke basic SOAP-over-HTTP-based service instances without using WSDL.

- **EndPoint URL:** This is the SOAP endpoint location at the service provider to which Liberty SOAP messages are sent. An example of this for the Employee Profile is [BASEURL]/services/IDSISEmployeeProfile. If the service instance exposes an endpoint that is different from the logically generated concrete WSDL, you must use the WSDL URI instead.

  A WSF service description endpoint cannot contain double-byte characters.

- **SOAP Action:** The SOAP action HTTP header required on HTTP-bound SOAP messages. This header can be used to indicate the intent of a SOAP message to the recipient.

WSDL Service Access Method: Specify the method used to access the WSDL service. WSDL (Web Service Description Language) describes the interface of a Web service.

- **Service Name Reference:** A reference name for the service.
- **WSDL URI:** Provides a URI to an external concrete WSDL resource containing the service description. URIs need to be constant across all implementations of a service to enable interoperability.

7 Click *OK*.

8 Update the Identity Server configuration.

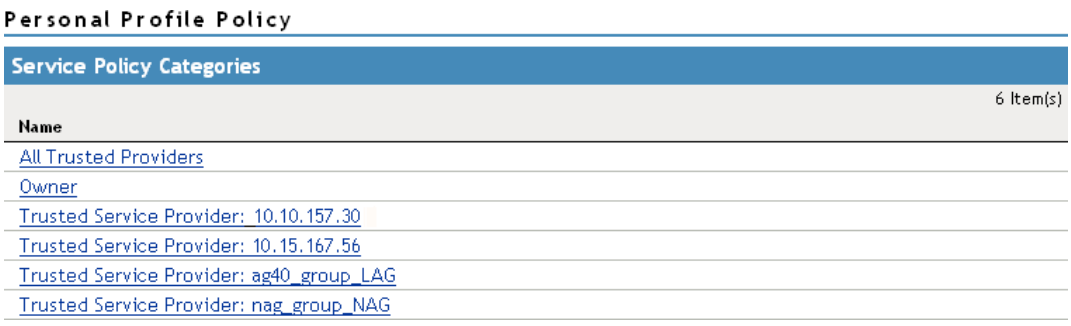
13.2.4 Editing Web Service Policies

Web Service policies are permission policies (query and modify) that govern how identity providers share end-user data with service providers. Administrators and policy owners (users) can control whether private information is always allowed to be given, never allowed, or must be requested.

As an administrator, you can configure this information for the policy owner, for specific service providers, or globally for all service providers. You can also specify what policies are displayed for the end user in the User Portal, and whether users are allowed to edit them.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.

2 Click the *Policy* link next to the service name.



3 Click the category you want to edit.

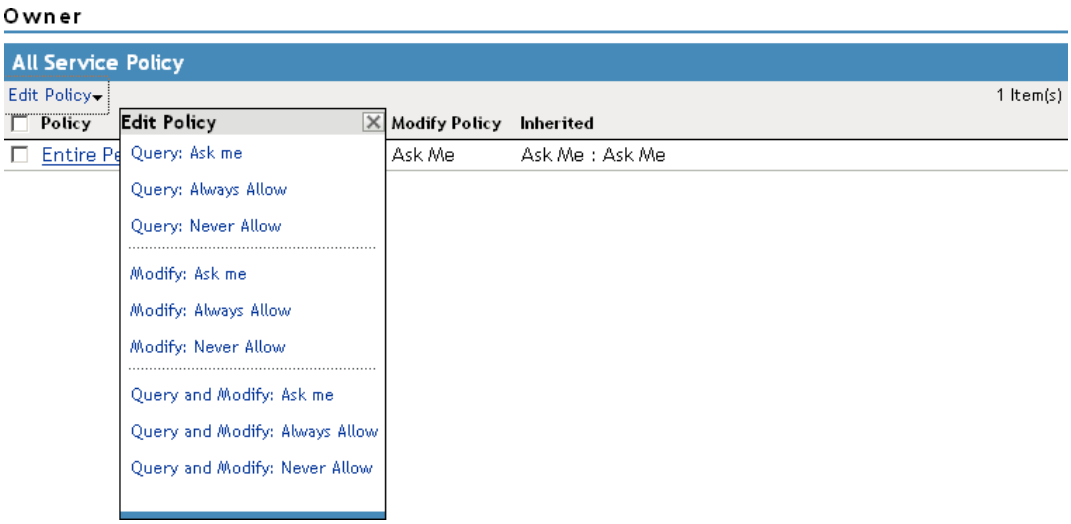
All Trusted Providers: Policies that are defined by the service provider's ability to query and modify the particular Liberty attributes or groups of attributes for the Web service. When All Trusted Providers permissions are established, and a service provider needs data, the system

first looks here to determine whether user data is allowed, never allowed, or must be asked for. If no solution is found in All Trusted Providers, the system examines the permissions established within the specific service provider.

Owners: Policies that limit the end user's ability to modify or query data from his or her own profile. The settings you specify in the *Owner* group are reflected on the My Profile page in the User Portal. Portal users have the authority to modify the data items in their profiles. The data items include Liberty and LDAP attributes for personal identity, employment, and any customized attributes defined in the Identity Server configuration. Any settings you specify in the Administration Console override what is displayed in the User Portal. Overrides are displayed in the *Inherited* column.

If you want the user to have Write permission for a given data item, and that data item is used in an LDAP Attribute Map, then you must configure the LDAP Attribute Map with Write permission.

4 On the All Service Policy page, select the policy's check box, then click *Edit Policy*.



This lets you modify the parent service policy attribute. Any selections you specify on this page are inherited by child policies.

Query Policy: Allows the service provider to query for the data on a particular attribute. This is similar to read access to a particular piece of data.

Modify Policy: Allows the service provider to modify a particular attribute. This is similar to write access to a particular piece of data.

Query and Modify: Allows you to set both options at once.

5 To edit child attributes of the parent, click the policy.

In the following example, child attributes are inheriting Ask Me permission from the parent *Entire Personal Identity* attribute. The *Postal Address* attribute, however, is modified to never allow permission for sharing.

Entire Personal Identity

Personal Identity

Edit Policy 12 Item(s)

Policy	Query Policy	Modify Policy	Inherited
Informal Name	Ask Me	Ask Me	Ask Me : Ask Me
Localized Informal Name	Ask Me	Ask Me	Ask Me : Ask Me
Entire Common Name	Ask Me	Ask Me	Ask Me : Ask Me
Entire Legal Identity	Ask Me	Ask Me	Ask Me : Ask Me
Employment Identity	Ask Me	Ask Me	Ask Me : Ask Me
Postal Addresses	Never Allow	Never Allow	Ask Me : Ask Me
Contact Profiles	Ask Me	Ask Me	Ask Me : Ask Me
Internet Identity	Ask Me	Ask Me	Ask Me : Ask Me

If you click the *Postal Address* attribute, you can see that all of its child attributes have inherited the *Never Allow* setting. You can specify different permission attributes for *Address Type* (for example), but the inherited policy still overrides changes made at the child level, as shown below.

Postal Addresses

Postal Addresses

Edit Policy 6 Item(s)

Policy	Query Policy	Modify Policy	Inherited
Address Type	Always Allow	Always Allow	Never Allow : Never Allow
NickName	Ask Me	Ask Me	Never Allow : Never Allow
Localized NickNames	Ask Me	Ask Me	Never Allow : Never Allow
Comment	Ask Me	Ask Me	Never Allow : Never Allow
Postal Address	Ask Me	Ask Me	Never Allow : Never Allow
Postal Addresses Extensions	Ask Me	Ask Me	Never Allow : Never Allow

The interface allows these changes in order to simplify switching between configurations if, for example, you want to remove an inherited policy.

Inherited: Specifies the settings inherited from the parent attribute policy, when you view a child attribute. In the User Portal, settings displayed under *Inherited* are not modifiable by the user. At the top-level policy in the User Portal, the values are inherited from the settings in the Administration Console. Thereafter, inheritance can come from the service policy or the parent data item's policy.

Ask Me: Specifies that the service provider requests from the user what action to take.

Always Allow: Specifies that the identity provider always allows the attribute data to be sent to the service provider.

Never Allow: Specifies that the identity provider never allows the attribute data to be sent to the service provider.

When a request for data is received, the Identity Server examines policies to determine what action to take. For example, if a service provider requires a postal address for the user, the Identity Server performs the following actions:

- Checks the settings specified in *All Service Providers*.
- If no solution is found, checks for the policy settings configured for the service provider.

6 Click *OK* until the Web Service Provider page is displayed.

7 Click *OK*, then update the Identity Server as prompted.

13.2.5 Create Web Service Type

This page allows you to create a Web service profile type. This is Step 1 of the Create Web Service Wizard. Access Manager comes with several Web service profiles, but if you have deleted a profile type, you can create it again.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider* > *New*.
2. Select the Web service type from the drop-down list, then click *Next*.
3. Continue with one of the following:
   - Section 13.2.1, "Modifying Service and Profile Details for Employee, Custom, and Personal Profiles," on page 297.
   - Section 13.2.2, "Modifying Details for Authentication, Discovery, LDAP, and User Interaction Profiles," on page 299.

13.3 Configuring Credential Profile Security and Display Settings

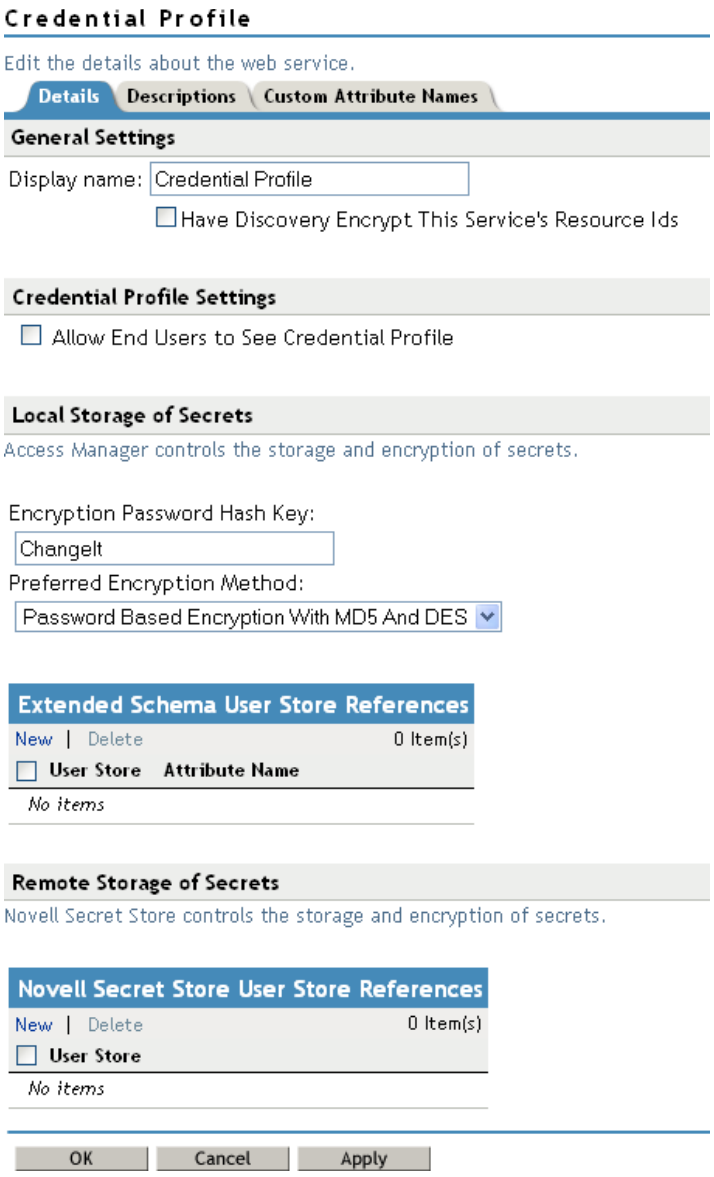
On the Credential Profile Details page, you can specify whether this profile is displayed for end users, and determine how you control and store encrypted secrets. You can store and access secrets locally, on remote eDirectory servers that are running Novell SecretStore, or on a user store that has been configured with a custom attribute for secrets.

For more information about storing encrypted secrets, see the following:

- For information on how to configure Access Manager for secrets, see Section 3.1.4, "Configuring a User Store for Secrets," on page 109.
- For general information about Novell SecretStore, see the *Novell SecretStore Administration Guide* (http://www.novell.com/documentation/secretstore33/pdfdoc/nssadm/nssadm.pdf).
- For information about creating shared secrets for Form Fill and Identity Injection policies, see "Creating and Managing Shared Secrets" in the *Novell Access Manager 3.1 SP2 Policy Guide*.

To configure the Credential Profile:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Providers*.
2. Click *Credential Profile*.



3 On the Credential Profile Details page, fill in the following fields as necessary:

Display name: The name you want to display for the Web service.

Have Discovery Encrypt This Service's Resource Ids: Specify whether the Discovery Service encrypts resource IDs. A resource ID is an identifier used by Web services to identify a user. The Discovery Service returns a list of resource IDs when a trusted service provider queries for the services owned by a given user. The Discovery Service has the option of encrypting the resource ID or sending it unencrypted. Encrypting resource IDs is disabled by default.

4 Under *Credential Profile Settings*, enable the following option if necessary:

Allow End Users to See Credential Profile: Specify whether to display or hide the Credential Profile in the Access Manager User Portal. Profiles are viewed on the My Profile page, where the user can modify his or her profile.

5 Specify how you want to control and store secrets:

5a To locally control and store secrets, configure the following fields:

Encryption Password Hash Key: (Required) Specify the password that you want to use as a seed to create the encryption algorithm. To increase the security of the secrets, we recommend that you change the default password to a unique alphanumeric value.

Preferred Encryption Method: Specify the preferred encryption method. Select the method that complies with your security model:

- **Password Based Encryption With MD5 and DES:** MD5 is an algorithm that is used to verify data integrity. Data Encryption Standard (DES) is a widely used method of data encryption using a private key.
- **DES:** Data Encryption Standard (DES) is a widely used method of data encryption using a private key. Like other private key cryptographic methods, both the sender and the receiver must know and use the same private key.
- **Triple DES:** A variant of DES in which data is encrypted three times with standard DES using two different keys.

5b Specify where to store secret data. (For more information about setting up a user store for secret store, see Section 3.1.4, "Configuring a User Store for Secrets," on page 109.)

- To have the secrets stored in the configuration database, do not configure the list in the *Extended Schema User Store References* section. You only need to configure the fields in Step 5a.
- To store the secrets in your LDAP user store, click *New* in *Extended Schema User Store References* and configure the following fields:

  User Store: Select a user store where secret data is stored.

  Attribute Name: Specify the LDAP attribute of the User object that can be used to store the secrets. When a user authenticates by using the user store specified here, the secret data is stored in an XML document of the specified attribute of the user object. This attribute should be a single-valued case ignore string that you have defined and assigned to the user object in the schema.
- To use Novell SecretStore to remotely store secrets, click *New* under *Novell Secret Store User Store References*.

  Click the user store that you have configured for SecretStore.

  Secure LDAP must be enabled between the user store and the Identity Server in order to add this user store reference.

5c Click *OK* twice.

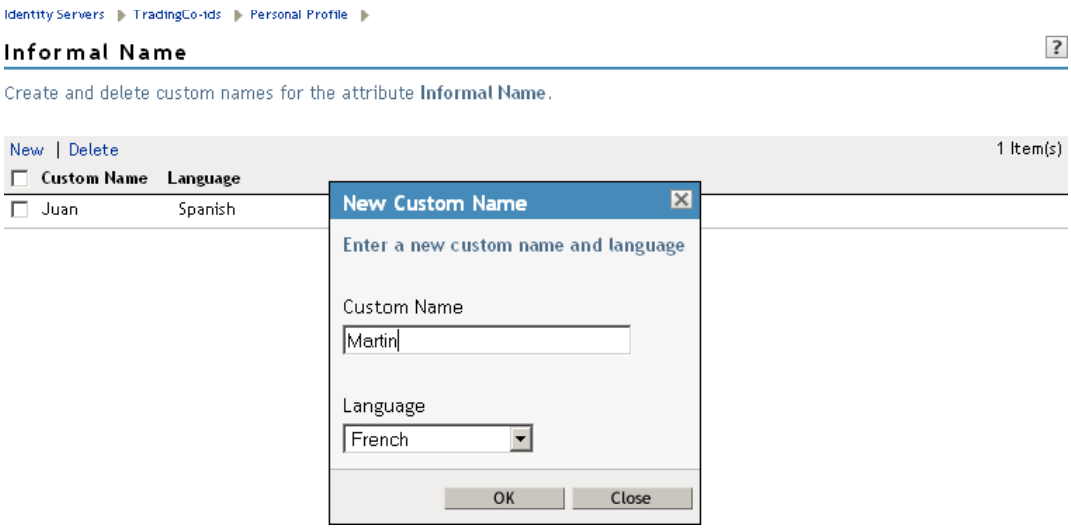
6 On the Identity Server page, update the Identity Server.

13.4 Customizing Attribute Names

You can change the display names of the attributes for the Credential, Custom, Employee, and Personal profiles. The customized names are displayed on the My Profile page in the User Portal. The users see the custom names applicable to their language. Custom Attributes are displayed on the My Profile page in the User Portal in place of the corresponding English attribute name when the language in the drop-down list is the accepted language of the browser.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider* > *[Profile]* > *Custom Attribute Names*.

2 Click the data item name to view the customized attribute names.



3 Click *New* to create a new custom name.

4 Type the name and select a language.

5 Click *OK*.

6 On the Custom Attribute Names page, click *OK*.

7 On the Web Service Provider page, click *OK*.

8 Update the Identity Server configuration on the Servers page.

13.5 Configuring the Web Service Consumer

The Web service consumer is the component within the identity provider that requests attributes from Web service providers. The identity provider and Web services consumer cooperate to redirect the user or resource owner to the identity provider, allowing interaction. You can configure an interaction service, which allows the identity provider to pose simple questions to a user. This service can be offered by trusted Web services consumers, or by a dedicated interaction service provider that has a reliable means of communication with the users.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Consumers*

The following general settings configure time limits and processing speed:

Protocol Timeout (seconds): Limits the time the transport protocol allows.

Provider Timeout (seconds): Limits the request processing at the Web service provider. This value must always be equal to or greater than the *Protocol Timeout* value.

Attribute Cache Enabled: A subsystem of the Web service consumer that caches attribute data that the Web service consumer requests. For example, if the Web service consumer has already requested a first name attribute from a Web service provider, the Web service consumer does not need to request the attribute again. This setting improves performance when enabled. However, you can disable this option to increase system memory.

2 Specify how and when the identity provider interacts with the user:

Always Allow Interaction: Allows interaction to take place between users and service providers.

Never Allow Interaction: Never allows interaction between users and service providers.

Always Allow Interaction for Permissions, Never for Data: Allows interaction for permissions, never for data.

Maximum Allowed Interaction Time: Specifies the allowed time (in seconds).

3 To specify the allowable methods that a Web service provider can use for user interaction, click one of the following options:

Redirect to a User Interaction Service: Allows the Web service consumer to redirect the user agent to the Web service provider to ask questions. After the Web service provider has obtained the information it needs, it can redirect the user back to the Web service consumer.

Call a Trusted User Interaction Service: Allows the Web service provider to trust the Web service consumer to act as proxy for the resource owner.

4 Under *Security Settings*, fill in the following fields:

WSS Security Token Type: Instructs the Web service consumer/requestor how to place the token in the security header as outlined in the Liberty ID-WSF Security Mechanisms.

Signature Algorithm: The signature algorithm to use for signing the payload.

5 Click *OK*, then update the Identity Server configuration as prompted.

13.6 Mapping LDAP and Liberty Attributes

You can create an LDAP attribute map or edit an existing one. To create an attribute map, you specify how single-value and multi-value data items map to single-value and multi-value LDAP attributes. A single-value attribute can contain no more than one value, and a multi-value attribute can contain more than one. An example of a single-value attribute might be a person's gender, and an example of a multi-value attribute might be a person's various e-mail addresses, phone numbers, or titles.

1 In the Administration Console, click *Devices* > *Identity Servers* >> *Edit* > *Liberty* > *LDAP Attribute Mapping*.

2 Select one of the following actions:

New: Allows you create an LDAP attribute mapping. Select from the following types:

- **One to One:** Maps a single Liberty attribute to a single LDAP attribute. See Section 13.6.1, "Configuring One-to-One Attribute Maps," on page 309.
- **Employee Type:** Maps the Employee Type attribute to an LDAP attribute, then maps the possible Liberty values to LDAP values. See Section 13.6.2, "Configuring Employee Type Attribute Maps," on page 312.
- **Employee Status:** Maps the Employee Status attribute to an LDAP attribute, then maps the possible Liberty values to LDAP values. See Section 13.6.3, "Configuring Employee Status Attribute Maps," on page 313.
- **Postal Address:** Maps the Postal Address attribute to either multiple LDAP attributes or a delimited LDAP attribute. See Section 13.6.4, "Configuring Postal Address Attribute Maps," on page 315.
- **Contact Method:** Maps the Contact Method attribute to multiple LDAP attributes. See Section 13.6.5, "Configuring Contact Method Attribute Maps," on page 316.

- **Gender:** Maps the Gender attribute to an LDAP attribute, then maps the possible Liberty values to LDAP values. See Section 13.6.6, “Configuring Gender Attribute Maps,” on page 318.
- **Marital Status:** Maps the Marital Status attribute to an LDAP attribute, then maps the possible Liberty values to LDAP values. See Section 13.6.7, “Configuring Marital Status Attribute Maps,” on page 319.

Delete: Deletes the selected mapping.

Enable: Enables the selected mapping.

Disable: Disables the selected mapping. When the mapping is disabled, the server does not load the definition. However, the definition is not deleted.

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

13.6.1 Configuring One-to-One Attribute Maps

A one-to-one map enables you to map single-value and multiple-value LDAP attribute names to standard Liberty attributes. A default one-to-one attribute map is provided with Access Manager, but you can also define your own.

An example of a one-to-one attribute map might be the single-valued Liberty attribute Common Name (CommonName) used by the Personal Profile that is mapped to the LDAP attribute givenName. You can further configure the various Liberty values to map to any LDAP attribute names that you use.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *LDAP Attribute Mapping* > *New* > *One to One*.

2 Configure the following fields:

Type: Displays the type of mapping you are modifying or creating:

Name: The name you want to give the map.

Description: A description of the map.

Access Rights: A drop-down menu that provides the broadest control for the page. If you set this to *Read/Write*, you can specify rights for individual data items.

In order for user provisioning to succeed, you must select *Read/Write* from the *Access Rights* drop-down menu for any maps that use an attribute during user provisioning.

User Stores: The user store that a map applies to. If a user logs into a user store that is not in the map’s user store list, that map is not used to read or write attributes for that user.

3 Use the following guidelines to configure the map:

- Mapping Personal Profile Single-Value Data Items to LDAP Attributes
- Mapping Personal Profile Multiple-Value Data Items to LDAP Attributes
- Mapping Employee Profile Single-Value Data Items to LDAP Attributes
- Mapping Employee Profile Multiple-Value Data Items to LDAP Attributes
- Mapping Custom Profile Single-Value Data Items to LDAP Attributes
- Mapping Custom Profile Multiple-Value Data Items to LDAP Attributes

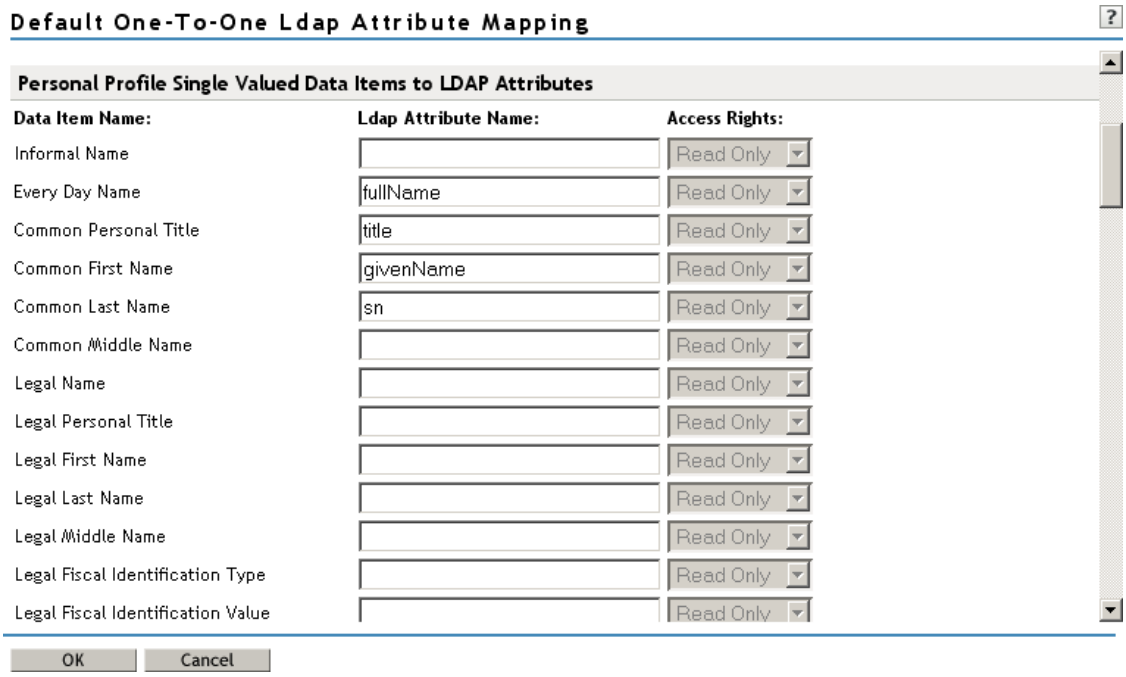
4 After you create the mapping, click *Finish*.

5 On the LDAP Attribute Mapping page, click *OK*.

6 Update the Identity Server.

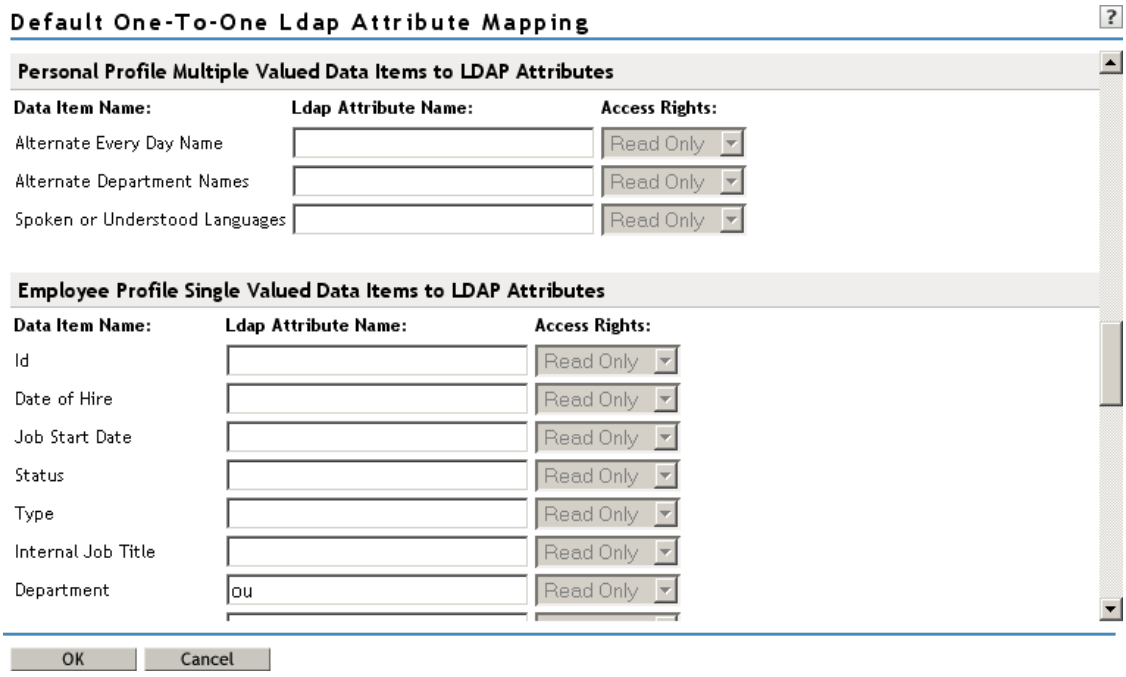
Mapping Personal Profile Single-Value Data Items to LDAP Attributes

The data items displayed are single-value Liberty Personal Profile attributes that you can map to the single-valued LDAP attributes that you have defined for your directory.



Mapping Personal Profile Multiple-Value Data Items to LDAP Attributes

Use the fields on this page to map multiple-value attributes from the Liberty Personal Profile to the multiple-value LDAP attributes you have defined for your directory. For example, you can map the Liberty attribute Alternate Every Day Name (AltCN) to the LDAP attribute you have defined for this purpose in your directory.



Mapping Employee Profile Single-Value Data Items to LDAP Attributes

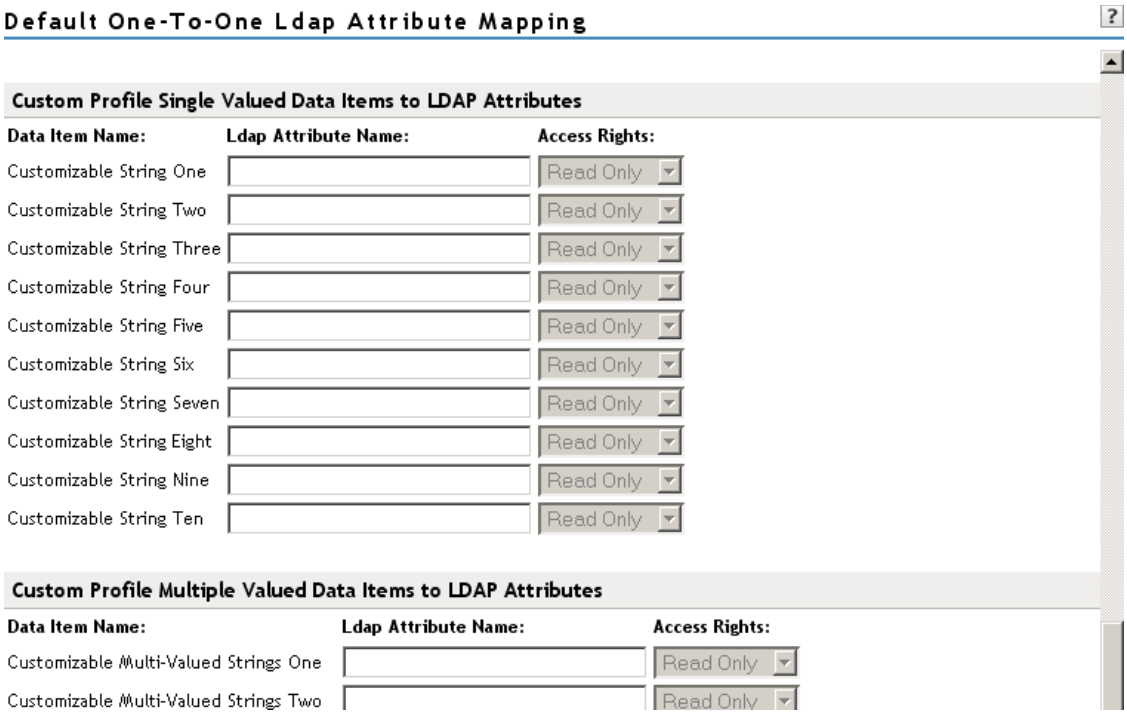
Map the Liberty Employee Profile single-value attributes to the LDAP attributes you have defined in your directory for entries such as ID, Date of Hire, Job Start Date, Department, and so on.

Mapping Employee Profile Multiple-Value Data Items to LDAP Attributes

Map the Liberty Employee Profile multiple-value attributes to the LDAP attributes you have defined in your directory.

Mapping Custom Profile Single-Value Data Items to LDAP Attributes

Map custom Liberty profile single-value attributes to LDAP attributes you have defined in your directory. These attributes are customizable strings associated with the Custom Profile.



Customizable String (1 - 10): The Custom Profile allows custom single-value and multiple-value attributes to be defined without using the Data Model Extension XML to extend a service's schema. To use a customizable attribute, navigate to the *Custom Attribute Names* tab on the Custom Profile Details page (see Section 13.4, "Customizing Attribute Names," on page 306). Use the page to customize the name of any of the predefined single-value or multiple-value customizable attributes in the Custom Profile. After you customize a name, you can use that attribute in the same way you use any other profile attribute.

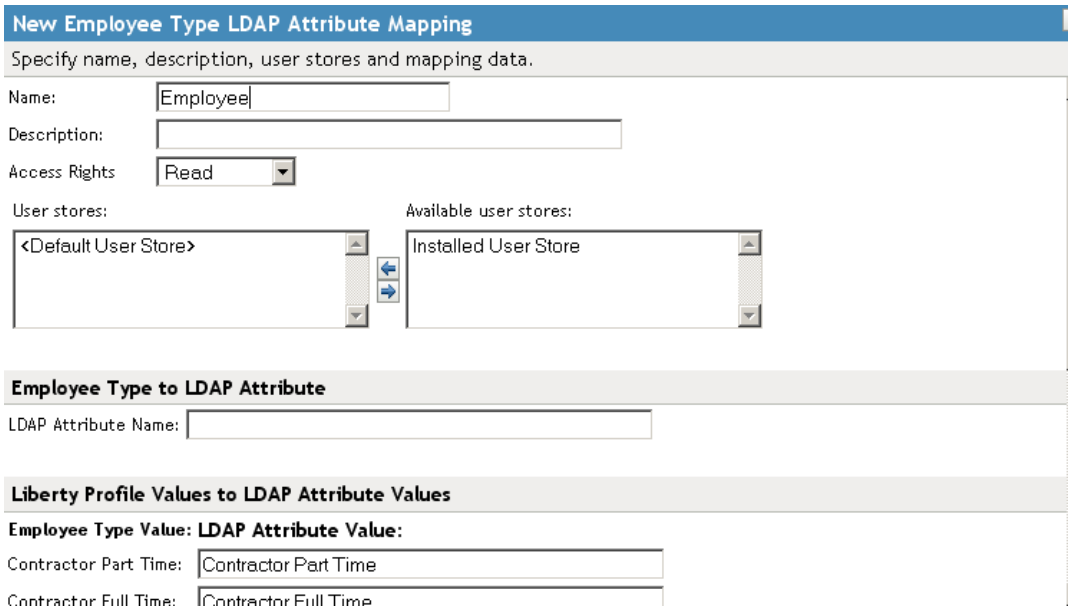
Mapping Custom Profile Multiple-Value Data Items to LDAP Attributes

Customizable Multi-Valued Strings (1 - 5): Similar to customizable strings for single-value attributes, except these attributes can have multiple values. Use this list of fields to map directory attributes that can have multiple values to multiple-value strings from the Custom Profile.

13.6.2 Configuring Employee Type Attribute Maps

You can map the LDAP attribute name and values to the Liberty profile values for Employee Type. This is an Employee Profile attribute. Examples of Liberty values appended to this attribute include Contractor Part Time, Contractor Full Time, Full Time Regular, and so on.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *LDAP Attribute Mapping* > *New* > *Employee Type*.



2 Configure the following fields:

Name: The name you want to give the map.

Description: A description of the map.

Access Rights: A drop-down menu that provide the broadest control for the page. If you set this to *Read/Write*, you can specify rights for individual data items.

In order for user provisioning to succeed, you must select *Read/Write* from the *Access Rights* drop-down menu for any maps that use an attribute during user provisioning.

User Stores: The user store that a map applies to. If a user logs into a user store that is not in the map's user store list, that map is not used to read or write attributes for that user.

3 In the *LDAP Attribute Name* field, type the LDAP attribute name that you want to map to the Liberty Employee Type attribute.

4 In the *LDAP Attribute Value* fields, type the predefined LDAP attribute values that you want to map to the *Liberty Employee Type* values.

These are the values that you want to store in the LDAP attribute for each given Liberty attribute value. The LDAP attribute map then maps the actual Liberty URI value, back and forth, to this supplied value.

5 Click *Finish*.

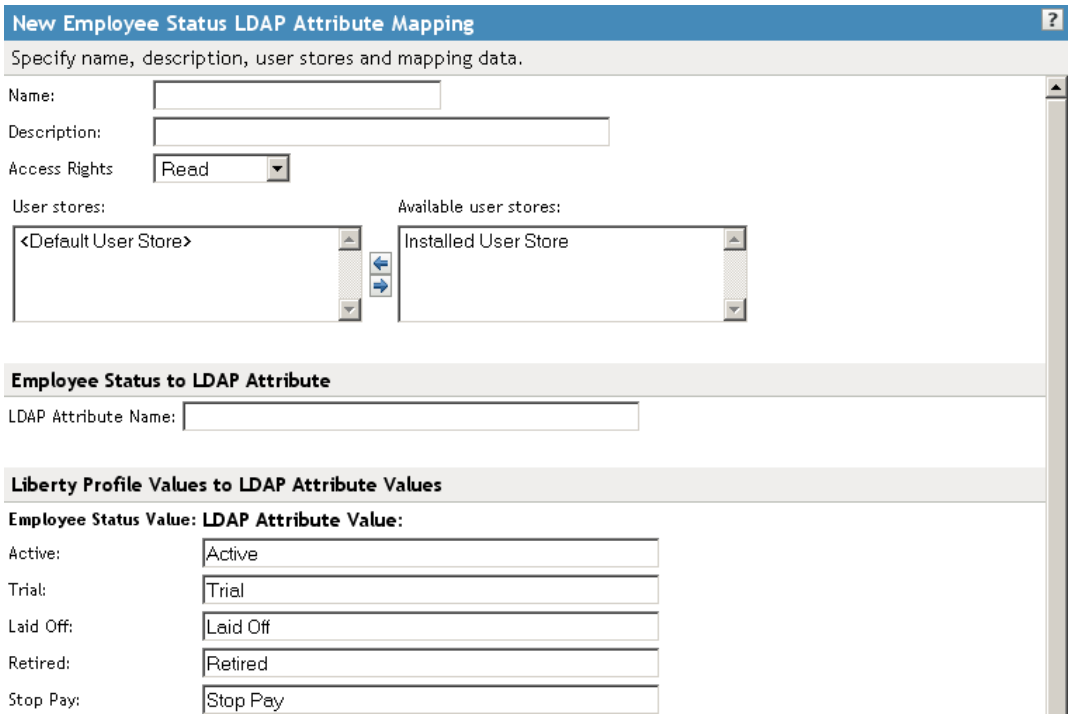
6 On the LDAP Attribute Mapping page, click *OK*.

7 Update the Identity Server.

13.6.3 Configuring Employee Status Attribute Maps

You can map the LDAP attribute name and values to the Liberty profile values for Employee Status. This is an Employee Profile attribute. Examples of the values appended to this Liberty attribute include Active, Trial, Retired, Terminated, and so on.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *LDAP Attribute Mapping* > *New* > *Employee Status*.



2 Configure the following fields:

Name: The name you want to give the map.

Description: A description of the map.

Access Rights: A drop-down menu that provide the broadest control for the page. If you set this to *Read/Write*, you can specify rights for individual data items.

In order for user provisioning to succeed, you must select *Read/Write* from the *Access Rights* drop-down menu for any maps that use an attribute during user provisioning.

User Stores: The user store that a map applies to. If a user logs into a user store that is not in the map's user store list, that map is not used to read or write attributes for that user.

3 In the *LDAP Attribute Name* field, type the LDAP attribute name that you want to map to the *Liberty Employee Status* element.

4 In the *LDAP Attribute Value* fields, type the predefined LDAP attribute values that you want to map to the *Liberty Employee Status* values.

These are the values that you want to store in the LDAP attribute for each given Liberty attribute value. The LDAP attribute map then maps the actual Liberty URI value, back and forth, to this supplied value.

5 Click *Finish*.

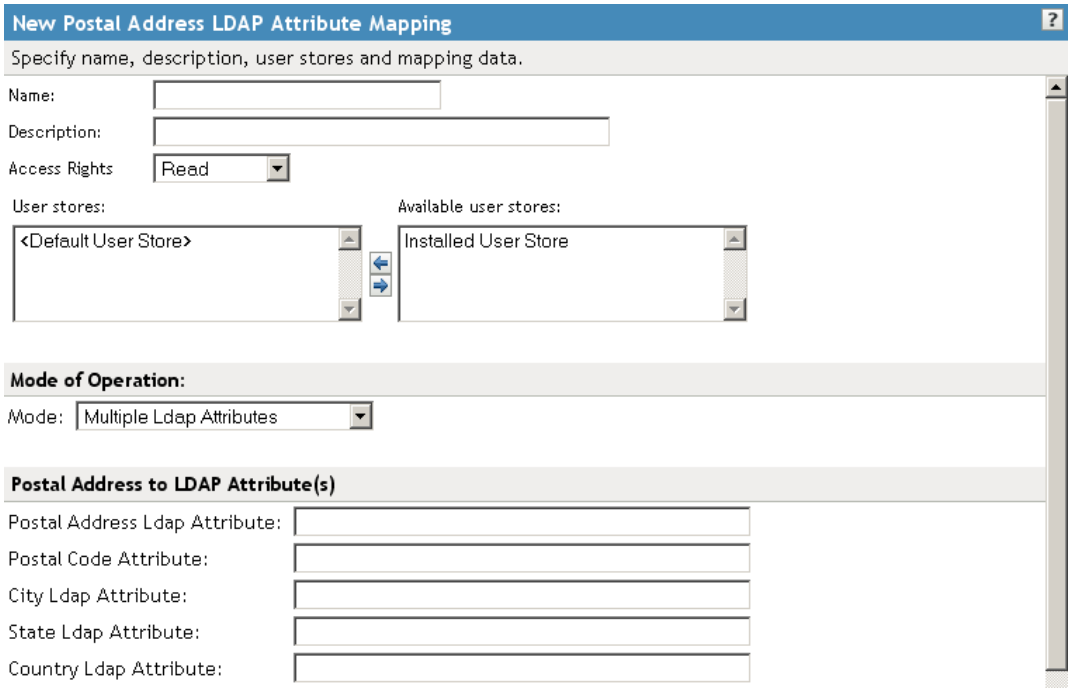
6 On the LDAP Attribute Mapping page, click *OK*.

7 Update the Identity Server.

13.6.4 Configuring Postal Address Attribute Maps

You can map the LDAP attribute name and values to the Liberty profile values for Postal Address. The PostalAddress element refers to the local address, including street or block with a house number, and so on. This is a Personal Profile attribute.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *LDAP Attribute Mapping* > *New* > *Postal Address*.



2 Configure the following fields:

Name: The name you want to give the map.

Description: A description of the map.

Access Rights: A drop-down menu that provide the broadest control for the page. If you set this to *Read/Write*, you can specify rights for individual data items.

In order for user provisioning to succeed, you must select *Read/Write* from the *Access Rights* drop-down menu for any maps that use an attribute during user provisioning.

User Stores: The user store that a map applies to. If a user logs into a user store that is not in the map's user store list, that map is not used to read or write attributes for that user.

3 In the *Mode* drop-down menu, select either *Multiple LDAP Attributes* or *Single Delimited LDAP Attributes*.

Multiple LDAP Attributes: Allows you to map multiple LDAP attributes to multiple Liberty Postal Address elements. When you select this option, the following Liberty Postal Address elements are displayed under the *Postal Address to LDAP Attributes* group. Type the LDAP attributes that you want to map to the Liberty elements.

- Postal Address
- Postal Code
- City

- State
- Country

Single Delimited LDAP Attributes: Allows you to specify one LDAP attribute that is used to hold multiple elements of a Liberty Postal Address in a single delimited value. When you select this option, the page displays the following fields:

- **Delimited LDAP Attribute Name:** The delimited LDAP attribute name you have defined for the LDAP postal address that you want to map to the Liberty Postal Address attribute.
- **Delimiter:** The character to use to delimit single-value entries. A $ sign is the default delimiter.

4. (Single Delimited LDAP Attributes mode) Under *One-Based Field Position in Delimited LDAP Attribute*, specify the order in which the information is contained in the string. Select 1 for the value that comes first in the string, 2 for the value that follows the first delimiter, etc.
5. (Multiple LDAP Attributes mode) Under *Postal Address Template Data*, fill in the following options:

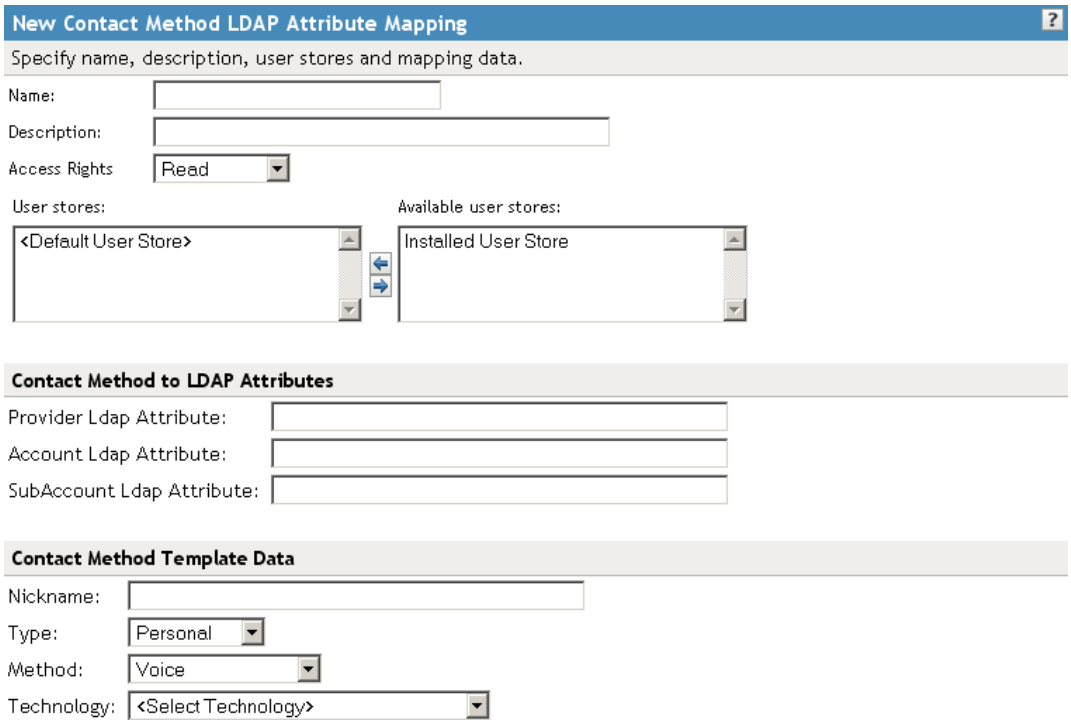
   Nickname: (Required) A Liberty element name used to identify the Postal Address object.

   Contact Method Type: Select the contact method type, such as *Domicile*, *Work*, *Emergency*, and so on.
6. Click *Finish*.
7. On the LDAP Attribute Mapping page, click *OK*.
8. Update the Identity Server.

13.6.5 Configuring Contact Method Attribute Maps

You can map the LDAP attribute you have defined for contact methods to the Liberty attribute Contact Method (MsgContact).

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *LDAP Attribute Mapping* > *New* > *Contact Method*.



2 Configure the following fields:

Name: The name you want to give the map.

Description: A description of the map.

Access Rights: A drop-down menu that provide the broadest control for the page. If you set this to *Read/Write*, you can specify rights for individual data items.

In order for user provisioning to succeed, you must select *Read/Write* from the *Access Rights* drop-down menu for any maps that use an attribute during user provisioning.

User Stores: The user store that a map applies to. If a user logs into a user store that is not in the map's user store list, that map is not used to read or write attributes for that user.

3 Under *Contact Method to LDAP Attributes*, fill in the following fields to map to the Liberty Contact Method attribute:

Provider LDAP Attribute: Maps to the Liberty attribute MsgProvider, which is the service provider or domain that provides the messaging service.

Account LDAP Attribute: Maps to the Liberty attribute MsgAccount, which is the account or address information within the messaging provider.

SubAccount LDAP Attribute: Maps to the Liberty MsgSubaccount, which is the subaccount within a messaging account, such as the voice mail box associated with a phone number.

4 Under *Contact Method Template Data*, specify the settings for the following Liberty attribute values:

Nickname: Maps to the Liberty attribute Nick, which is an informal name for the contact.

Type: Maps to the Liberty attribute MsgType (such as Mobile, Personal, or Work).

Method: Maps to the Liberty MsgMethod (such as Voice, Fax, or E-mail).

Technology: Maps to the Liberty attribute MsgTechnology (such as Pager, VOIP, and so on).

5 Click *Finish*.

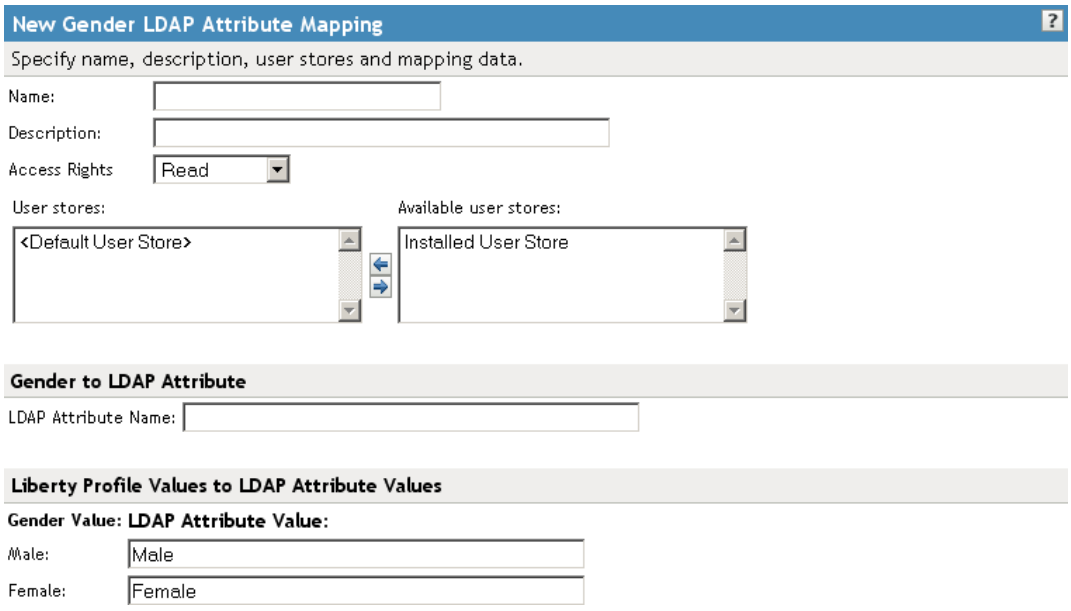
6 On the LDAP Attribute Mapping page, click *OK*.

7 Update the Identity Server.

13.6.6 Configuring Gender Attribute Maps

You can map the LDAP attribute name and values to the Liberty profile values for the Gender attribute. You can use gender to differentiate between people with the same name, especially in countries where national ID numbers cannot be collected. This is a Personal Profile attribute.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *LDAP Attribute Mapping* > *New* > *Gender*.



2 Configure the following fields:

Name: The name you want to give the map.

Description: A description of the map.

Access Rights: A drop-down menu that provide the broadest control for the page. If you set this to *Read/Write*, you can specify rights for individual data items.

In order for user provisioning to succeed, you must select *Read/Write* from the *Access Rights* drop-down menu for any maps that use an attribute during user provisioning.

User Stores: The user store that a map applies to. If a user logs into a user store that is not in the map's user store list, that map is not used to read or write attributes for that user.

3 In the *LDAP Attribute Name* field, type the LDAP attribute name that you want to map to the Liberty element Gender.

4 In the *LDAP Attribute Value* fields, type the predefined LDAP attribute values that you want to map to the Gender values.

These are the values that you want to store in the LDAP attribute for each given Liberty attribute value. The LDAP attribute map then maps the actual Liberty URI value, back and forth, to this supplied value.

5 Click *Finish*.

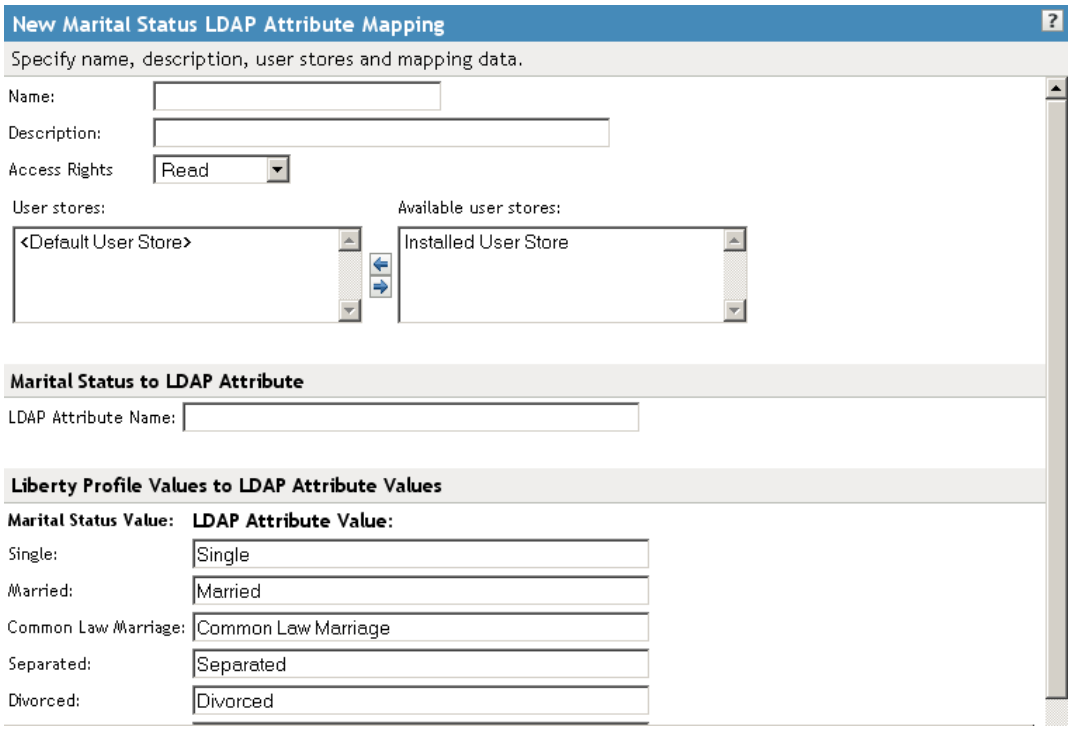
6 On the LDAP Attribute Mapping page, click *OK*.

7 Update the Identity Server.

13.6.7 Configuring Marital Status Attribute Maps

You can map the LDAP marital status attribute to the Liberty attribute. The Liberty Marital Status (MaritalStatus) element includes appended values such as single, married, divorced, and so on. For example, `urn:liberty:id-sis-pp:maritalstatus:single`. This is a Personal Profile attribute.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *LDAP Attribute Mapping* > *New* > *Marital Status*.



2 Configure the following fields:

Name: The name you want to give the map.

Description: A description of the map.

Access Rights: A drop-down menu that provide the broadest control for the page. If you set this to *Read/Write*, you can specify rights for individual data items.

In order for user provisioning to succeed, you must select *Read/Write* from the *Access Rights* drop-down menu for any maps that use an attribute during user provisioning.

User Stores: The user store that a map applies to. If a user logs into a user store that is not in the map's user store list, that map is not used to read or write attributes for that user.

3 In the *LDAP Attribute Name* field, type the LDAP attribute name that you want to map to the Liberty element Marital Status (MaritalStatus).

4 In the *LDAP Attribute Value* fields, type the predefined LDAP attribute values that you want to map to the MaritalStatus values.

These are the values that you want to store in the LDAP attribute for each given Liberty attribute value. The LDAP attribute map then maps the actual Liberty URI value, back and forth, to this supplied value.

5. Click *Finish*.
6. On the LDAP Attribute Mapping page, click *OK*.
7. Update the Identity Server.

Maintaining an Identity Server

14

Server maintenance involves tasks that you perform after you have configured the server. Maintenance includes monitoring the health of the servers, configuring logging, replacing certificates, monitoring statistics, and so on.

14.1 Managing an Identity Server

The Identity Servers page is the starting point for managing Identity Servers. Most often, you use this page to stop and start servers, and to assign servers to Identity Server configurations. An Identity Server cannot operate until you have assigned it to an Identity Server configuration.

1 In the Administration Console, click *Devices* > *Identity Servers*.

2 On the *Servers* tab, you can perform the following functions by clicking the server’s check box, then clicking any of the following options:

New Cluster: Creates a new cluster configuration. See Section 1.1.1, “Creating a Cluster Configuration,” on page 16.

Start: Starts the selected server. (See Section 14.1.2, “Restarting the Identity Server,” on page 323.)

Stop: Stops the selected server. (See Section 14.1.2, “Restarting the Identity Server,” on page 323.)

Refresh: Refreshes the server list.

Actions: Enables you to perform the following tasks:

- **Assign to Cluster:** Enables you to assign a server to a cluster configuration. See Section 1.1.2, “Assigning an Identity Server to a Cluster Configuration,” on page 21 for more information.
- **Remove from Cluster:** Enables you to remove one or more servers from a configuration. See Section 1.1.6, “Removing a Server from a Cluster Configuration,” on page 25 for more information.
- **Delete:** Deletes the selected server.

  IMPORTANT: The system does not allow you to delete an Identity Server that is started. You must first stop the server, then delete it. This removes the configuration object from the configuration store on the Administration Console. To remove the server software from the machine where it was installed, you must run the uninstall script on the server machine.
- **Update Health from Server:** Performs a health check for the device.

This page also displays links in the following columns:

Column	Description
Name	Lists Identity Server and cluster configuration names.
Status	Lists the status of each configuration. **Current:** Indicates that the server is using the latest configuration data. If you change a configuration, the system displays an *Update* or *Update All* link. **Update:** A link to update an Identity Server’s configuration data without stopping the server. **Update All:** A link displayed for cluster configurations. This lets you update all the Identity Servers in a cluster to use the latest configuration data, with options to include logging and policy settings. For more information about the update process, see Section 14.1.1, “Updating an Identity Server Configuration,” on page 322.
Health	Lists the health of each configuration and each server.
Alerts	Displays the Alerts page, where you can monitor and acknowledge server alerts.
Commands	Displays the Command Status page.
Statistics	Displays the Server Statistics page and allows you to view the server statistics. See Section 14.6, “Monitoring Identity Server Statistics,” on page 337.
Configuration	Lists the Identity Server configuration to which this server belongs.

14.1.1 Updating an Identity Server Configuration

Whenever you change an Identity Server configuration, the system prompts you to update the configuration. An *Update Servers* status is displayed under the *Status* column on the Servers page. You must click *Update Servers* to update the configuration so that your changes take effect.

When you click this link, it sends a reconfigure command to all servers that use the configuration. The servers then begin the reconfiguration process. This process occurs without interruption of service to users who are currently logged in.

When you update a configuration, the system blocks inbound requests until the update is complete. The server checks for any current requests being processed. If there are such requests in process, the server waits five seconds and tests again. This process is repeated three times, waiting up to fifteen seconds for these requests to be serviced and cleared out. After this period of time, the update process begins. Any remaining requests might have errors.

During the update process, all settings are reloaded with the exception of the base URL. In most cases, user authentications are preserved; however, there are conditions during which some sessions are automatically timed out. These conditions are:

- A user logged in via an authentication contract that is no longer valid. This occurs if an administrator removes a contract or changes the URI that is used to identify it.
- A user logged in to a user store that is no longer valid. This occurs if you remove a user store or change its type. Changing the LDAP address to a different directory is not recommended, because the system does not detect the change.
- A user received authentication from an identity provider that is no longer trusted. This occurs if you remove a trusted identity provider or if the metadata for the provider changed.

Additionally, if you remove a service provider from an identity provider, the identity provider removes the provided authentication to that service provider. This does not cause a timeout of the session.

Changes to the SAML and Liberty protocol profiles can result in the trusted provider having outdated metadata for the Identity Server being reconfigured. This necessitates an update at the other provider and might cause unexpected behavior until that occurs.

1. In the Administration Console, click *Devices* > *Identity Servers*.
2. Click *Update* or *Update All*.

   These options are only available when you have made changes that require a server update.

14.1.2 Restarting the Identity Server

Starting and stopping an Identity Server terminates active user sessions. These users receive a prompt to log in again unless you have configured session failover (see Section 1.1.4, “Configuring Session Failover,” on page 22).

1. In the Administration Console, click *Devices* > *Identity Servers*, then select the Identity Server to stop.
2. Click *Stop*.
3. Wait for the *Command Status* to change from *Pending* to *Complete*.
4. Select the Identity Server, then click *Start*.
5. When the *Command Status* changes to *Complete*, click *Refresh*.

   The status icon of the Identity Server should turn green.

14.2 Editing Server Details

You can edit server details, such as the server name and port. You can also access the other server management tabs from this page.

1. In the Administration Console, click *Devices* > *Identity Servers*, then click the server name.
2. To edit the information, click *Edit*.
3. Modify the following fields as necessary:

   Name: The name of the Identity Server. Names must be alphanumeric and can include spaces, hyphens, and underscores.

   Management IP Address: The IP address of the Identity Server. Changing server IP addresses is not recommended and causes the server to stop reporting. See "Changing the IP Address of an Identity Server" in the *Novell Access Manager 3.1 SP2 Administration Console Guide*.

   Port: The Identity Server port used for management.

   Location: The location of the Identity Server.

   Description: A description of the Identity Server.
4. To save your changes, click *OK*. Otherwise, click *Cancel*.

14.3 Configuring Component Logging

You can enable and configure how the system performs logging. Logging is the main tool you use for debugging the Identity Server configuration. All administrative and end-user actions and events are logged to a central event log. This allows easy access to this information for security and operational purposes. Additionally, the log system provides the ability to monitor ongoing activities such as identity provider authentication activity, up-time of the system, and so on. File logging is not enabled by default.

Identity Servers and Embedded Service Providers use these logging features. If you change or enable logging, you must update the Identity Server configuration and restart the Embedded Service Providers in order to apply the changes. When you disable logging, you must also restart the Embedded Service Providers.

This section describes the following about component logging:

14.3.1 Enabling Component Logging

File logging records the actions that have occurred. For example, you can configure Identity Server logging to list every request made to the server. With log file analysis tools, you can get a good idea of where visitors are coming from, how often they return, and how they navigate through a site. The content logged to file logging can be controlled by specifying logger levels and by enabling statistics logging.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Logging*.

2 The following options are available for component logging in the *File Logging* section:

- **Enabled:** Enables file logging for this server and its associated Embedded Service Providers.
- **Echo To Console:** Copies the Identity Server XML log file to `/var/opt/novell/tomcat5/logs/catalina.out` (Linux), to `/Program Files/Novell/Tomcat/logs/stdout.log` (Windows Server 2003), or to `/Program Files (x86)/Novell/Tomcat/logs/stdout.log` (Windows Server 2008). You can download the file from *Auditing* > *General Logging*.

  For the Embedded Service Providers, the log file location depends upon the device:

  - For an Access Gateway Appliance, a Linux Access Gateway Service, or an ESP-enabled SSL VPN server, this sends the messages to the `catalina.out` file of the device.
  - For a Windows Access Gateway Service, this sends messages to the `stdout.log` file of the device.
- **Log File Path:** Specifies the path that the system uses to save the Identity Server XML log file. The default path is *`tomcat application directory`*`/web-inf/logs`.

  If you change this path, you must ensure that the user associated with configuring the identity or service provider has administrative rights to the Tomcat application directory in the new path.

  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. In a mixed platform environment, you must use the default path.
- **Maximum Log Files:** Specifies the maximum number of Identity Server XML log files to leave on the machine. After this value is reached, the system deletes log files, beginning with the oldest file. You can specify *Unlimited*, or values of 1 through 200. 10 is the default value.
- **File Wrap:** Specifies the frequency (hour, day week, month) for the system to use when closing an XML log file and creating a new one. The system saves each file based on the time you specify and attaches the date and/or time to the filename.
- **GZip Wrapped Log Files:** Uses the GZip compression utility to compress logged files. The log files that are associated with the *GZip* option and the *Maximum Log Files* value are stored in the directory you specify in the *Log File Path* field.

3 In the *Component File Logger Levels* section, you can specify the logging sensitivity for the following:

Application: Logs system-wide events, except events that belong to a specific subsystem.

Liberty: Logs events specific to the Liberty IDFF protocol and profiles.

SAML 1: Logs events specific to the SAML1 protocol and profiles.

SAML 2: Logs events specific to the SAML2 protocol and profiles.

STS: Logs events specific to the STS protocol.

CardSpace: Logs events specific to the CardSpace protocol.

WS Federation: Logs events specific to the WS Federation protocol.

Web Service Provider: (Liberty) Logs events specific to fulfilling Web service requests from other Web service consumers.

Web Service Consumer: (Liberty) Logs all events specific to requesting Web services from a Web service provider.

Use the drop-down menu to categorize logging sensitivity. Higher logging levels also include the lower levels in the log.

- **Off:** Turns off component file logging for the selected item.
- **Severe:** Logs serious failures that can cause system processing to not proceed.
- **Warning:** Logs potential failures, but the impact on execution is minimal. Warnings indicate that you should be aware that this event is happening and might want to make a configuration change to avoid it.
- **Info:** Logs informational events. No execution or data impact occurred.
- **Verbose:** Logs static configuration information. The system logs any configuration errors under one of the primary three levels: Severe, Warning, and Info.
- **Debug:** Includes all of the preceding levels.

4 (Optional) Enable statistics logging:

When statistics logging is enabled, the system periodically sends the system statistics, in string format, to the current file logger. Statistical data (such as counts, levels, and so on) are included in the file log.

4a In the *Statistics Logging* section, select *Enabled*.

4b In the *Log Interval* field, specify the time interval in seconds that statistics are logged.

5 For information on configuring Novell Audit Logging, see Section 14.7, “Enabling Identity Server Audit Events,” on page 345.

6 Click *OK*.

7 Update the Identity Server.

8 Restart the Embedded Service Providers on the devices, in order to apply the changes.

When you disable component logging, you need to update the Identity Servers and restart the Embedded Service Providers.

14.3.2 Managing Log File Size

On Windows, you need to manually monitor the size of the log files. On Linux, the logrotate daemon manages the log files located in the following directories:

```
/var/opt/novell/tomcat5/logs
/opt/volera/roma/logs/
```

The logrotate daemon has been configured to scan the files in these directories once a day. It rolls them over when they have reached their maximum size and deletes the oldest version when the maximum number of copies have been created.

If you want to modify this behavior, see the following files in the `/etc/logrotate.d` directory:

```
novell-tomcat5
novell-devman
```

For information about the parameters in these files, see the documentation for the logrotate daemon.

14.4 Configuring Session-Based Logging

The session-based logging feature allows the administrator to enable file logging for an individual user. In production environments, this has the following value:

- Debug logging can be turned on for an individual user rather than all users. The potential size of logged data usually prohibits an administrator from turning on debug logging for all users.
- All logged messages for this user are directed to a single file. Administrators do not need to sort through the various log files to follow the activity of the user.
- Isolating the problem and finding the cause is limited to the user who is experiencing the problem.
- Enabling session-based logging does not require a configuration change to the Identity Server, and thus does not require updating the Identity Server.

The following user scenario explains how this feature could be used in a production environment

1. A user notices a problem and calls the help desk.
2. The help desk operator questions the users and concludes that the problem is caused by either a Novell Identity Server or an Embedded Service Provider.
3. The operator has been granted the rights to create logging tickets, and uses the User Portal to create a logging ticket for the user.
4. The operator sends the logging ticket password and the URL to access the logging ticket class to the user.
5. The user clicks the URL and enters the logging ticket password.

   This marks the current session as “active for logging” and adds a small icon to the top right of the page, which makes the session logging feature visible to the user.
6. Using the same browser window, the user duplicates the problem behavior.
7. The operator can then access the data that was logged just for this user and analyze the cause of the behavior.

To enable session-based logging, the following tasks need to be completed:

- Section 14.4.1, “Creating the Administrator Class, Method, and Contract,” on page 327
- Section 14.4.2, “Creating the Logging Session Class, Method, and Contract,” on page 329
- Section 14.4.3, “Enabling Basic Logging,” on page 330
- Section 14.4.4, “Responding to an Incident,” on page 330

14.4.1 Creating the Administrator Class, Method, and Contract

The IDP Administrator class, method, and contract control who has the rights to create a logging ticket. You need to know the DNs of the operators who are going to be responding to the users who are experiencing problems.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local*.

2 To create the class:

2a Click *Classes*.

2b Click *New*, then specify the following values:

Display name: `IDP Administrator`

Java class: `Other`

Java class path: com.novell.nidp.authentication.local.IDPAdministratorClass

2c Click *Next*, then click *Finish*.

3 To create the method:

3a Click *Methods*.

3b Click *New*, then specify the following values:

Display name: `IDP Administrator Method`

Class: `IDP Administrator`

Identifies user: Deselect this option.

User Stores: Select the user stores that contain your operators, then move them to the list of User Stores.

3c In the *Properties* section, click *New*, then specify the following to create an IDP Administrator:

Property Name: `Administrator1`

The Property Name must begin with Administrator; append a value to this so that each property has a unique value.

Property Value: `cn=jdoe,o=users`

The Property Value must be the DN of an operator in the user stores you selected in Step 3b. Use LDAP typed comma notation for the DN.

3d Repeat Step 3c for each IDP Administrator you require.

You can return to this method to add or remove IDP Administrators, when responsibilities change.

3e Click *Finish*.

4 To create the contract:

4a Click *Contracts*.

4b Click *New*, then specify the following values:

Display name: `IDP Administrator Contract`

URI: `urn:novell:nidp:admin:contract`

Methods: Move the *IDP Administrator Method* to the Methods list.

Leave all other fields with their default values.

4c Click *Next*, then specify the following values for the authentication card:

ID: `IDPAdmin`

Text: `IDP Administrator`

Image: Select an image from the list, such as the IDP Administrator image that was created for this type of contract.

Show Card: Deselect this option.

4d Click *Finish*.

5 Continue with "Creating the Logging Session Class, Method, and Contract" on page 329.

14.4.2 Creating the Logging Session Class, Method, and Contract

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Local*.
2. To create the class:
   - **2a** Click *Classes*.
   - **2b** Click *New*, then specify the following values:

     Display name: `Logging Session`

     Java class: `Other`

     Java class path: com.novell.nidp.authentication.local.LogTicketClass
   - **2c** Click *Nex*t, then click *Finish*.
3. To create the method:
   - **3a** Click *Methods*.
   - **3b** Click *New*, then specify the following values:

     Display name: `Logging Session Method`

     Class: `Logging Session`

     Identifies user: Deselect this option.

     User Stores: Select the user stores that contain the users that potentially can experience problems, then move them to the list of User Stores.
   - **3c** Click *Finish*.
4. To create the contract:
   - **4a** Click *Contracts*.
   - **4b** Click *New*, then specify the following values:

     Display name: `Logging Session Contract`

     URI: `urn:novell:nidp:loggingsession:contract`

     Methods: Move the *Logging Session Method* to the *Methods* list.

     Leave all other fields with their default values.
   - **4c** Click *Next*, then specify the following values for the authentication card:

     ID: `LogSession`

     Text: `Logging Session`

     Image: Select an image from the list, for example the Session Logging image that was created for this type of contract.

     Show Card: Deselect this option.
   - **4d** Click *Finish*.
5. Click *OK*, then update the Identity Server.
6. Continue with "Enabling Basic Logging" on page 330.

14.4.3 Enabling Basic Logging

For session-based logging to function, logging on the Identity Server must be enabled. However, you do not need to select what is logged. The Logging Ticket enables the appropriate components and levels when an incident occurs.

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit*.
2. Click *Logging*, then specify the following:

   File Logging: Enable this option.

   Echo To Console: Enable this option.

   No other options need to be enabled. The *Component File Logger Levels* can be left in their default state of off.
3. Click *OK*, then update the Identity Server.

   This completes the configuration. You now need to wait for a user to report a problem. For information on using this feature to respond to a problem, see "Responding to an Incident" on page 330.

14.4.4 Responding to an Incident

The following sections explain how to use the feature when a user reports a problem:

Creating a Logging Ticket

These steps are performed by an IDP Administrator when a user reports a problem:

1. Log in to the Identity Server, using the credentials of an IDP Administrator.

   If the base URL of the Identity Server is https://idp.amlab.net:8443/nidp, enter the following URL:

   ```
   https://idp.amlab.net:8443/nidp/app
   ```
2. Change to the Logging Ticket page by specifying the following URL:

   ```
   https://idp.amlab.net:8443/nidp/app/login?id=IDPAdmin
   ```

   The *id* specified in the URL must match the ID you specified for the ID of the IDP Administrator Contract. See Step 4c of Section 14.4.1, "Creating the Administrator Class, Method, and Contract," on page 327.

   If you logged in with the credentials of an IDP Administrator, an *Administrator* tab appears.
3. To create a ticket for the user, click the *Administrator* tab.

   3a Click *New*.

   3b Specify the following:

   Ticket: Specify a name for ticket.

   You must share this name with the user who reported the problem.

   Ticket Good For: Select a time limit for the ticket, from one minute through one year.

When selecting a time limit, consider the following:

- When a ticket expires, logging is automatically stopped. If you know that user is experiencing a problem that prevents the user from logging out, you might want to create a ticket with a short time limit.
- If the user does not log out (just closes the browser window or the problem closes it), the session remains in the list of logged sessions. After 10 minutes of inactivity, the session is closed and the lock on the log file is cleared. As long as the log file is locked, no other application can read the file.

Ticket Log Level: Select the level of information to log, from severe-only messages to debug.

Log to Console: Select to log the messages to the user's file and to the console.

- If you have set up logging for session-based logging (see "Enabling Basic Logging" on page 330), then this allows you see the messages in the `catalina.out` or `stdout.log` file.
- If you have enabled Component File Logger Levels, selecting this option can create duplicate entries in the `catalina.out` or `stdout.log` file.

3c Click *Create*.

4 Create a URL that uses the following format:

`https://<base_URL>/nidp/app/login?id=<LogSession>`

Replace <*base_URL*> with the base URL of your Identity Server, including the port. Make sure the port agrees with the HTTP scheme (either http or https).

Replace <*LogSession*> with the ID you specified for the authentication card when defining the Logging Session contract.

IMPORTANT: The id is the ID of the authentication card of the Logging Session contract (see Step 4c of Section 14.4.2, "Creating the Logging Session Class, Method, and Contract," on page 329). It is not the name of the ticket you just created.

If the base URL of the Identity Server is https://idp.amlab.net:8443/nidp and the ID for the authentication card is LogSession, create the following URL:

`https://idp.amlab.net:8443/nidp/app/login?id=LogSession`

5 Send the URL of the LogSession card and the name of the ticket to the user.

Enabling a Logging Session

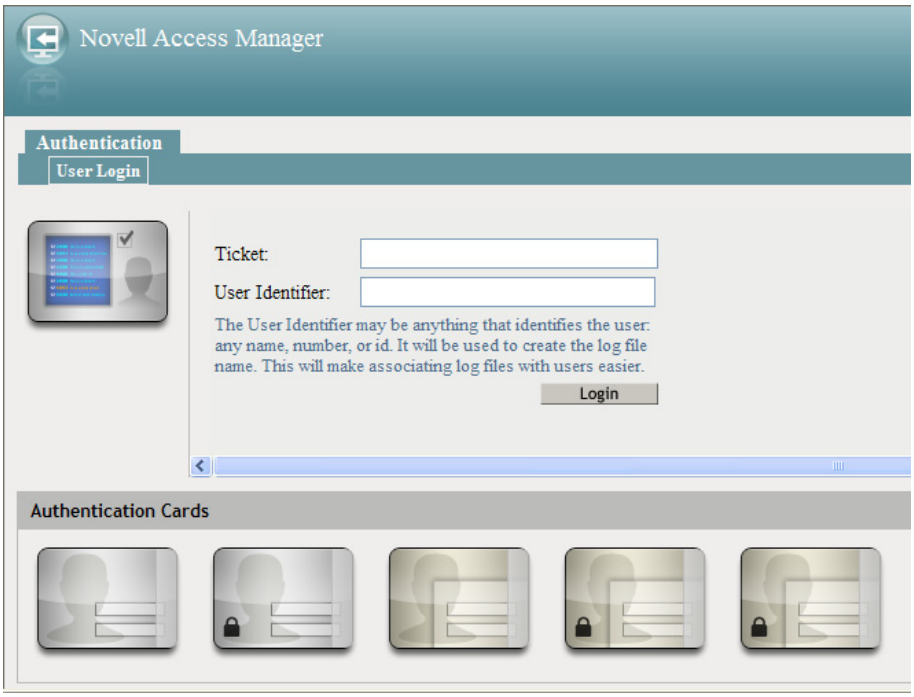
These steps are performed by the user. The URL needs to be sent to the user, with the ID and ticket values that were specified in "Creating a Logging Ticket" on page 330.

1 Open a browser and enter the log session URL sent by the help desk.

If the URL does not display a page that prompts for the ticket name, check the value of the id string. The id must be set to the ID of the authentication card of the Logging Session contract.

Instead of sending the user a URL, you can enable the *Show Card* option for the Logging Session card. When you do this, all users can see it. You need to decide if this is acceptable.

When the Show Card option is enabled, the login page looks similar to the following:



2 When prompted, enter the following:

Ticket: Specify the ticket name that the help desk sent.

User Identifier: Specify a value that the help desk associates with you as a user. The identifier must be less than 33 characters and contain only alphanumeric characters.

3 Click *Login*.

This login creates the logging session.

4 Enter your name and password, then click *Login*.

This login authenticates you to the Identity Server.

5 In the same browser window, enter the URL of the resource that is causing the problem.

6 Perform any other actions necessary to create the problem behavior.

7 Log out and send your user identifier to the help desk.

Viewing the Log File

These steps are performed by someone who has had Access Manager training and understands the significance of the messages in the log files. This can be an IDP Administrator or a specialist.

1 On the Identity Server, change to the Tomcat log directory.

Linux: `/var/opt/novell/tomcat5/webapps/nidp/WEB-INF/logs`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\WEB-INF\logs`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF\logs`

2 Open the file that begins with the user identifier to which a session ID is appended.

If the user does not log out (just closes the browser window or the problem closes it), the session remains in the list of logged sessions. After 10 minutes of inactivity, the session is closed and the lock on the logging file is cleared. As long as the file is locked, no other application can read the file.

When a ticket expires, logging is stopped automatically. If you know that user is experiencing a problem that prevents the user from logging out, you might want to create a ticket with a short time limit.

3 (Conditional) If the user was experiencing a problem with an Embedded Service Provider, change to the Tomcat log directory on the device:

Linux: `/var/opt/novell/tomcat5/webapps/nesp/WEB-INF/logs`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nesp\WEB-INF\logs`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nesp\WEB-INF\logs`

4 Open the file with the same user identifier and session ID.

5 After solving the problem, delete the file from each Identity Server in the cluster and each Access Gateway in the cluster.

14.5 Monitoring the Health of an Identity Server

14.5.1 Health States

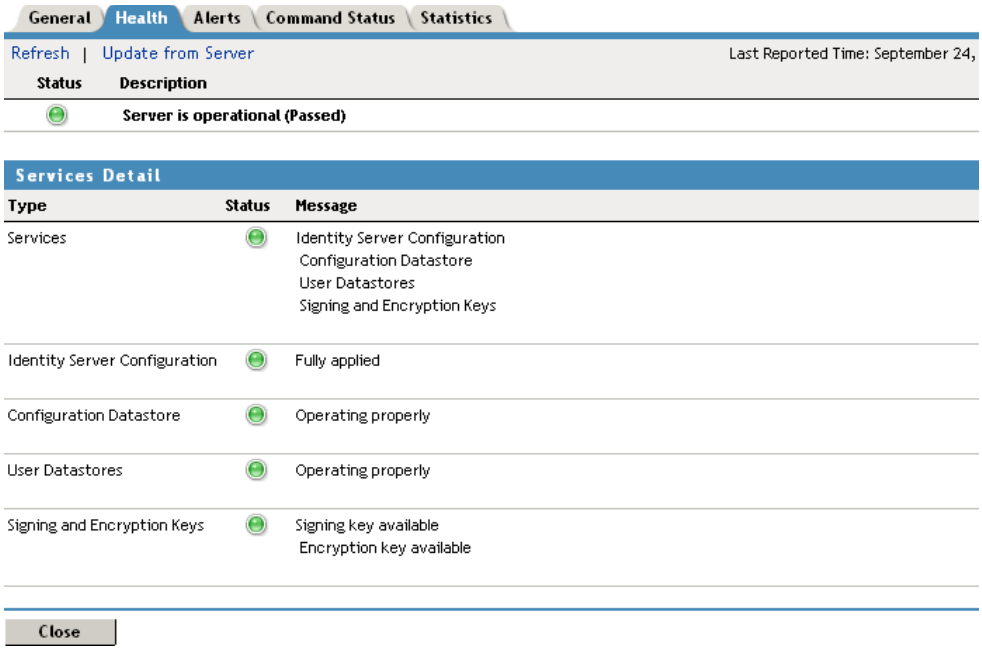
The Health page displays the current status of the server. The following states are possible:

Icon	Description
	A green status indicates that the server has not detected any problems
	A green status with a yellow diamond indicates that the server has not detected any problems but the configuration isn't completely up-to-date because commands are pending.
	A green status with a red x indicates that the server has not detected any problems but that the configuration might not be what you want because one or more commands have failed.
	A red status with a bar indicates that the server has been stopped.
	A white status with disconnected bars indicates that the server is not communicating with the Administration Console.
	A yellow status indicates that the server might be functioning sub-optimally because of configuration discrepancies.
	A yellow status with a question mark indicates that the server has not been configured.
	A red status with an x indicates that the server configuration might be incomplete or wrong, that a dependent service in not running or functional, or that the server is having a runtime problem.

14.5.2 Viewing the Health Details of an Identity Server

To view detailed health status information for an Identity Server:

1 In the Administration Console, click *Devices* > *Identity Servers* > *[Name of Server]* > *Health*.



The status icon is followed by a description that explains the significance of the current state. For more information about the icons, see Section 14.5.1, “Health States,” on page 333.

2 To ensure that the information is current, select one of the following:

- Click *Refresh* to refresh the page with the latest health available from the Administration Console.
- Click *Update from Server* to send a request to the Identity Server to update its status information. This can take a few minutes.

3 Examine the *Services Detail* section that displays the status of each service. For an Identity Server, this includes information such as the following:

Status Category	If not healthy
Status: Indicates whether the Identity Server is online and operational.	Verify whether the Identity Server has been stopped or is not configured. Also verify that network problems are not interfering with communications between the Identity Server and the Administration Console.
Services: Indicates the general health of all configured services.	If one service is unhealthy, this category reflects that status. See the particular service that also displays an unhealthy status.
Identity Server Configuration: Indicates the status of the configuration.	Configure the Identity Server or assign the server to a configuration. See Chapter 1, “Configuring an Identity Server,” on page 15.
Configuration Datastore: Indicates the status of the installed configuration datastore.	You might need to restart Tomcat or reinstall the Administration Console. If you have a backup Administration Console, you can restore it. See “Backing Up and Restoring” in the *Novell Access Manager 3.1 SP2 Administration Console Guide*. If you don’t have a backup, you can try repairing the configuration datastore. See “Repairing the Configuration Datastore” in the *Novell Access Manager 3.1 SP2 Administration Console Guide*. If you want to convert a secondary console to your primary console, see “Converting a Secondary Console into a Primary Console” in the *Novell Access Manager 3.1 SP2 Administration Console Guide*.
User Datastores: Indicates whether the Identity Server can communicate with the user stores, authenticate as the admin user, and find the search context.	Ensure that the user store is operating and configured correctly. You might need to import the SSL certificate for communication with the Identity Server. See Section 3.1, “Configuring Identity User Stores,” on page 104.
Signing, Encryption and SSL Connector Keys: Indicates whether these keystores contain valid a key.	Click *Identity Servers* > *Edit* > *Security* and replace any missing or expired keys.
System Incoming and Outgoing HTTP Requests: Appears when throughput is slow. This health check monitors incoming HTTP requests, outgoing HTTP requests on the SOAP back channel, and HTTP proxy requests to cluster members. If one or more requests remain in the queue for over 2 minutes, this health check appears.	Verify that all members of the cluster have sufficient bandwidth to handle requests. If a cluster member is going down, the problem resolves itself as other members of the cluster are informed that the member is down. If a cluster member is slow because it doesn’t have enough physical resources (speed or memory) to handle the load, upgrade the hardware.

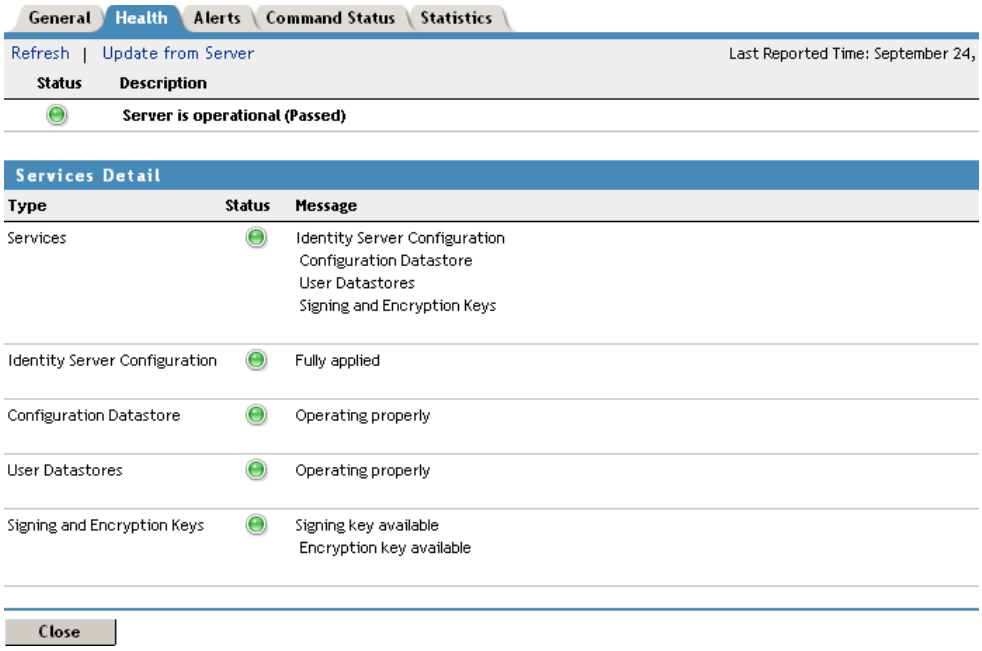
Status Category	If not healthy
SSL Communication: Indicates whether SSL communication is operating correctly. This health check appears only when the SSL communication check fails.	Check SSL connectivity. Check for expired SSL certificates.
Audit Logging Server: Indicates whether the audit agent is functioning and able to log events to the auditing server. Auditing must be enabled on the Identity Server to activate this health check (click *Devices* > *Identity Servers* > *Edit* > *Logging*).	Check the network connection between the Identity Server and the auditing server. See "Troubleshooting Novell Audit" (http://www.novell.com/documentation/novellaudit20/novellaudit20/data/al0lh30.html).

4 Click *Close*.

14.5.3 Viewing the Health Details of a Cluster

The health page displays the current health of the cluster.

1 In the Administration Console, click *Devices* > *Identity Servers* > *[Name of Cluster]* > *Health*.



The status icon is followed by a description that explains the significance of the current state. For more information about the icons, see Section 14.5.1, "Health States," on page 333.

2 To ensure that the information is current, click *Refresh* to refresh the page with the latest health available from the Administration Console.

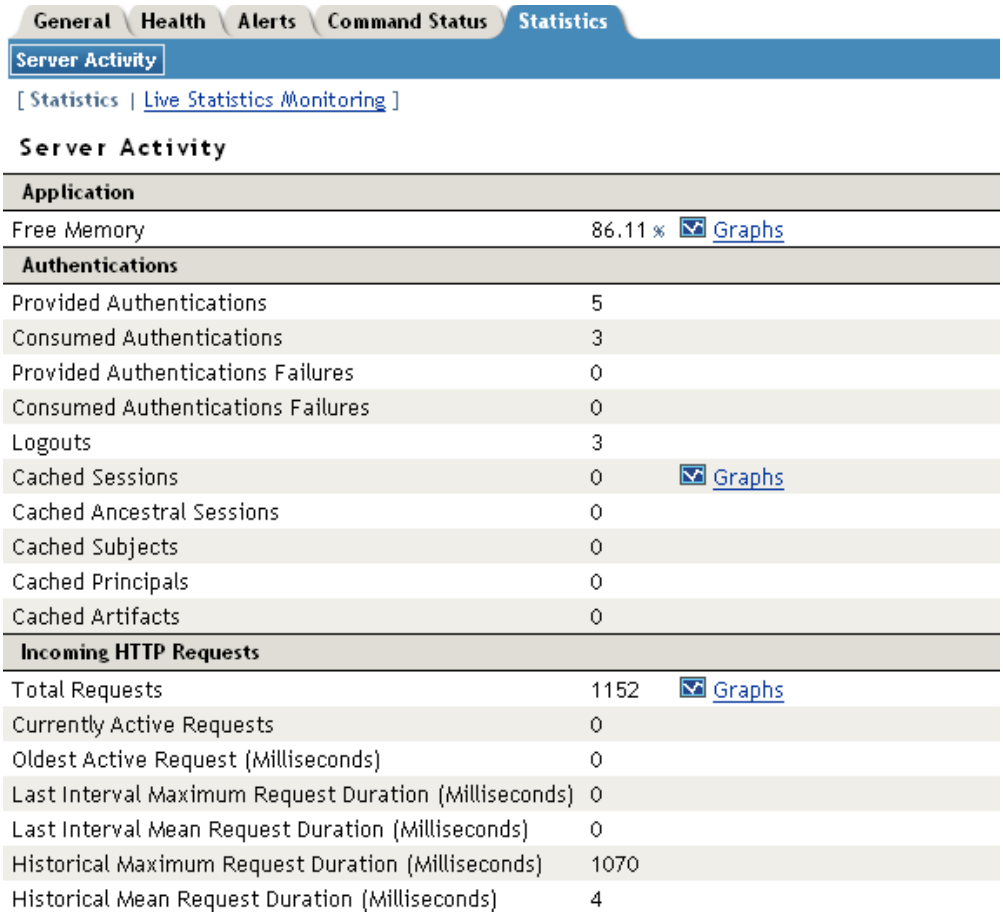
3 To view health details about a specific member of the cluster, click the server's health icon.

14.6 Monitoring Identity Server Statistics

The Statistics page allows you to monitor the amount of data and the type of data the Identity Server is processing. You can specify the intervals for the refresh rate and, where allowed, view graphic representations of the activity.

1 In the Administration Console, choose *Devices* > *Identity Servers*.

2 In the *Statistics* column, click *View*.



3 Click either of the following options:

Statistics: Select this option to view the statistics as currently gathered. The page is static and the statistics are not updated until you click *Live Statistics Monitoring*.

Live Statistics Monitoring: Select this option to view the statistics as currently gathered and to have them refreshed at the rate specified in the *Refresh Rate* field.

4 Review the following statistics:

- Application
- Authentications
- Incoming HTTP Requests
- Outgoing HTTP Requests
- Liberty

- SAML 1.1
- SAML 2
- WSF (Web Services Framework)
- Clustering
- LDAP

5 Click *Close* to return to the Servers page.

14.6.1 Application

Statistic	Description
Free Memory	The percentage of free memory available to the JVM (Java Virtual Machine). Click *Graphs* to view memory usage for a specific unit of time (1 hour, 1 day, 1 week, 1 month, 6 months, or 12 months). The Value axis displays the percentage of memory that is free for the selected time period.

14.6.2 Authentications

Statistic	Description
Provided Authentications	The number of successful provided authentications given out to external entities since the Identity Server was started.
Consumed Authentications	The number of successful consumed authentications since the Identity Server was started.
Provided Authentication Failures	The number of failed provided authentications given out to external entities since the Identity Server was started.
Consumed Authentication Failures	The number of failed consumed authentications since the Identity Server was started.
Logouts	The number of explicit logouts performed by users. This does not include logouts where an inactive session was destroyed.
Cached Sessions	The number of currently active cached user sessions. This represents the number of users currently logged into the system; however, if a single person has two browser windows open on the same client and if that person performed two distinct authentications, then that person has two user sessions. Click *Graphs* to view the number of cached sessions for a specific unit of time (1 hour, 1 day, 1 week, 1 month, 6 months, or 12 months). The Value axis displays the number of cached sessions. If no sessions have been cached, the value axis is not meaningful.
Cached Ancestral Sessions	The number of cached ancestral session IDs. An ancestral session ID is created during the failover process. When failover occurs, a new session is created to represent the previous session. The ID of the previous session is called an “ancestral session ID,” and it is retained for subsequent failover operations.

Statistic	Description
Cached Subjects	The number of current cached subject objects. Conceptually, the cached subjects are identical to the cached principals.
Cached Principals	The number of current cached principal objects. A principal can be thought of as a single directory user object. Multiple users can log in using a single directory user object, in which case multiple cached sessions would exist sharing a single cached principal.
Cached Artifacts	The number of current cached artifact objects. During authentication, an artifact is generated that maps to an assertion. This cache holds the artifact to assertion mapping until the artifact resolution request is received. Under normal operations, artifacts are resolved within milliseconds of being placed in this cache.

14.6.3 Incoming HTTP Requests

Incoming HTTP requests are divided into three categories: active, interval, and historical. As soon as a request is complete, it is placed into the interval category. The interval represents the last 60 seconds of processed requests. At the completion of the 60-second interval, all requests in the interval category are merged into the historical category.

Statistic	Description
Total Requests	The total number of incoming HTTP requests that have been processed since the Identity Server was started. Click *Graphs* to view the number of requests for a specific unit of time (1 hour, 1 day, 1 week, 1 month, 6 months, or 12 months). The Value axis displays the number of requests for the selected time period.
Currently Active Requests	The number of currently active incoming HTTP requests.
Oldest Active Request (Milliseconds)	The age of the oldest currently active incoming HTTP request.
Last Interval Maximum Request Duration (Milliseconds)	The age of the longest incoming HTTP requests that was processed during the last 60-second interval.
Last Interval Mean Request Duration (Milliseconds)	The mean age of all incoming HTTP request that were processed during the last 60-second interval.
Historical Maximum Request Duration (Milliseconds)	The age of the longest incoming HTTP request that was processed since the Identity Server was started.
Historical Mean Request Duration (Milliseconds)	The mean age of all incoming HTTP requests that were processed since the Identity Server was started.

14.6.4 Outgoing HTTP Requests

Outgoing HTTP requests are divided into three categories: active, interval, and historical. As soon as a request is complete, it is placed into the interval category. The interval represents the last 60 seconds of processed requests. At the completion of the 60-second interval, all requests in the interval category are merged into the historical category.

Statistic	Description
Total Requests	The total number of outgoing HTTP requests that have been processed since the Identity Server was started. Click *Graphs* to view the number of requests for a specific unit of time (1 hour, 1 day, 1 week, 1 month, 6 months, or 12 months). The Value axis displays the number of requests for the selected time period.
Currently Active Requests	The number of currently active outgoing HTTP requests.
Oldest Active Request (Milliseconds)	The age of the oldest currently active outgoing HTTP request.
Last Interval Maximum Request Duration (Milliseconds)	The age of the longest outgoing HTTP request that was processed during the last 60-second interval.
Last Interval Mean Request Duration (Milliseconds)	The mean age of all outgoing HTTP requests that were processed during the last 60-second interval.
Historical Maximum Request Duration (Milliseconds)	The age of the longest outgoing HTTP request that was processed since the Identity Server was started.
Historical Mean Request Duration (Milliseconds)	The mean age of all outgoing HTTP requests that were processed since the Identity Server was started.

14.6.5 Liberty

Statistic	Description
Liberty Federation	The number of Liberty protocol federations performed since the Identity Server was started.
Liberty De-Federations	The number of Liberty protocol defederations performed since the Identity Server was started.
Liberty Register-Names	The number of Liberty protocol register names performed since the Identity Server was started.

14.6.6 SAML 1.1

Statistic	Description
SAML1.1 Attribute Queries	The number of SAML 1.1 protocol attribute queries performed since the Identity Server was started.

14.6.7 SAML 2

Statistic	Description
SAML2 Attribute Queries	The number of SAML 2 protocol attribute queries performed since the Identity Server was started.
SAML2 Federations	The number of SAML 2 protocol federations performed since the Identity Server was started.
SAML2 Defederations	The number of SAML 2 protocol defederations performed since the Identity Server was started.
SAML2 Register-Names	The number of SAML 2 protocol register names performed since the Identity Server was started.

14.6.8 WSF (Web Services Framework)

Statistic	Description
Personal Profile Service Queries	The number of Liberty IDSIS Personal Profile Web Service queries performed since the Identity Server was started.
Personal Profile Service Modifies	The number of Liberty IDSIS Personal Profile Web Service changes performed since the Identity Server was started.
Employee Profile Service Queries	The number of Liberty IDSIS Employee Profile Web Service queries performed since the Identity Server was started.
Employee Profile Service Modifies	The number of Liberty IDSIS Employee Profile Web Service changes performed since the Identity Server was started.
Custom Profile Service Queries	The number of Novell Custom Profile Web Service queries performed since the Identity Server was started.
Custom Profile Service Modifies	The number of Novell Custom Profile Web Service changes performed since the Identity Server was started.
Credential Profile Service Queries	The number of Novell Credential Profile Web Service queries performed since the Identity Server was started.
Credential Profile Service Modifies	The number of Novell Credential Profile Web Service changes performed since the Identity Server was started.
Authentication Profile Service Queries	The number of Novell Authentication Profile Web Service queries performed since the Identity Server was started.
Authentication Profile Service Modifies	The number of Novell Authentication Profile Web Service changes performed since the Identity Server was started.
LDAP Profile Service Queries	The number of Novell LDAP Profile Web Service queries performed since the Identity Server was started.
LDAP Profile Service Modifies	The number of Novell LDAP Profile Web Service changes performed since the Identity Server was started.
Constant Profile Service Queries	The number of Novell Constant Profile Web Service queries performed since the Identity Server was started.

Statistic	Description
Discovery Service Queries	The number of Liberty Discovery Web Service queries performed since the Identity Server was started.
Discovery Service Modifies	The number of Liberty Discovery Web Service changes performed since the Identity Server was started.
Redirected Interaction Service Requests	The number of Liberty User Interaction Redirection Profile requests performed since the Identity Server was started.
Trusted Interaction Service Requests	The number of Liberty User Interaction Trusted Service Profile requests performed since the Identity Server was started.
Client of Redirected Interaction Service Requests	The number of Liberty User Interaction Redirection Profile requests initiated as a client since the Identity Server was started.
Client of Trusted Interaction Service Requests	The number of Liberty User Interaction Trusted Service Profile requests initiated as a client since the Identity Server was started.
Data Location LDAP	The number of attempts to use LDAP as a data location for a query or a modify of any Web Service since the Identity Server was started.
Data Location LDAP Aggregation	The number of attempts to use LDAP as a data location for aggregation of a query or a modify of any Web Service since the Identity Server was started.
Data Location User Profile	The number of attempts to use the User Profile object as a data location for a query or a modify of any Web Service since the Identity Server was started. A User Profile object is a directory object stored in the Identity Server's configuration datastore.
Data Location User Profile Aggregation	The number of attempts to use the User Profile object as a data location for aggregation of a query or a modify of any Web Service since the Identity Server was started. A User Profile object is a directory object stored on the Identity Server's configuration datastore.
Data Location Remote	The number of attempts to use the Remote location as a data location for a query or a modify of any Web Service since the Identity Server was started. A Remote location includes Pushed Attributes and External Services.
Data Location Pushed Attributes	The number of attempts to use the Pushed Attributes as a remote data location for a query or a modify of any Web Service since the Identity Server was started.
Data Location Pushed Attributes Aggregation	The number of attempts to use the Pushed Attributes as an remote data location for aggregation of a query or a modify of any Web Service since the Identity Server was started.
Data Location External Service	The number of attempts to use an External Service as a remote data location for a query or a modify of any Web Service since the Identity Server was started. An External Service is where the same Web Service exists on an external Service Provider and a call can be made to request data from the service.

14.6.9 Clustering

An authoritative server is the cluster member that holds the authentication information for a given user session. For a request associated with a given session to be processed, it must be routed ("proxied") to the authoritative cluster member. If an L4 switch causes a request to go to a non-authoritative cluster member, that cluster member proxies the request to the authoritative cluster member.

When a request is received, a cluster member uses multiple means to determine which cluster member is the authoritative server for the request. It looks for a parameter on the query string of the URL indicating the authoritative server. It looks for an HTTP cookie, indicating the authoritative server. If these do not exist, the cluster member examines the payload of the HTTP request to determine the authoritative server. Payload examinations result in immediate identification of the authoritative server or a user session ID or user identity ID that can be used to locate the authoritative server.

If a user session ID or user identity ID is found, the ID is broadcast to all cluster members asking which member is the authoritative server for the given ID. The authoritative server receives the broadcast message, determines that it indeed holds the given session or user, and responds accordingly.

The higher the number of proxied requests, the lower the performance of the entire system. Furthermore, the higher the number of payload examinations and ID broadcasts, the lower the performance of the entire system. If these numbers are high, verify the configuration of the L4 switch. Make sure that the session persistence option is enabled, which allows clients to be directed to the same Identity Server after they have established a session.

Statistic	Description
Currently Active Proxied Requests	The number of currently active proxied HTTP requests.
Total Proxied Requests	The total number of proxied requests that have been processed since the Identity Server was started. A request becomes a proxied request when the request is sent first to a non-authoritative machine.
Total Non-Proxied Requests	The total number of non-proxied requests that have been processed since the Identity Server was started. A request becomes a non-proxied request when the request is sent first to the authoritative machine.
Authoritative Server Obtained from URL Parameter	The total number of authoritative servers identified by using the parameter from the URL query string since the Identity Server was started.
Authoritative Server Obtained from Cookie	The total number of authoritative servers identified by using the HTTP cookie since the Identity Server was started.
Payload Examinations	The total number of attempted payload examinations to identify the authoritative server since the Identity Server was started.
Successful Payload Examinations	The total number of successful payload examinations to identify the authoritative server since the Identity Server was started.
Identity ID Broadcasts	The total number of attempted Identity ID Broadcasts to identify the authoritative server since the Identity Server was started.

Statistic	Description
Successful Identity ID Broadcasts	The total number of successful Identity ID Broadcasts to identify the authoritative server since the Identity Server was started.
Session ID Broadcasts	The total number of attempted Session ID Broadcasts to identify the authoritative server.
Successful Session ID Broadcasts	The total number of successful Session ID Broadcasts to identify the authoritative server since the Identity Server was started.

14.6.10 LDAP

Statistic	Description
Connections Created	The total number of LDAP connections created since the Identity Server was started. This count is a sum of all connections created to all replicas of the configuration datastore and all user stores.
Connections Destroyed	The total number of LDAP connections destroyed since the Identity Server was started. This count is a sum of all connections destroyed on all replicas of the configuration datastore and all user stores.
Connections Reused	The total number of times an LDAP connection was reused for a subsequent administrative task since the Identity Server was started.
Connections Shared Between Pools	The total number of times an LDAP connection count has been shared between connection pools since the Identity Server was started. Each LDAP replica contains two connection pools: the user connection pool and the administration connection pool. ◆ User connections are used to authenticate users and they are created and immediately destroyed. ◆ Administration connections are persisted in the pool and reused for administrative tasks. Each pool has a maximum number of current connections it is allowed to hold at any one time. Initially, the number of allowed connections is allocated evenly between the two pools. If a much greater demand is detected for one pool over the other, then the pools reallocate their maximum number of connections, increasing one pool's maximum by one and decreasing the other pool's maximum by one. When this happens, it is said that the pool “shared” a connection with the other pool.
User Store Replica Restarts	The number of times that a user store replica became unavailable so that a restart was necessary since the Identity Server was started. A user store restart is attempted once every minute.
Successful User Store Replica Restarts	The number of times that a user store replica restart was successfully completed since the Identity Server was started.
User Store Replica Restart Retries	The number of times that a user store replica restart failed and was put back into “wait mode” to try again in one minute since the Identity Server was started.
Currently Active Connection Waits	The current number of user threads waiting for an LDAP connection to become available.

Statistic	Description
Connection Waits	The number of times that a user thread was required to wait for an LDAP connection to become available since the Identity Server was started. A wait would be required if the maximum number of connections allocated to the associated connection pool were all currently in use by other threads.
Connection Waits Aborted Due To Timeout	The number of times that an LDAP connection wait terminated because of the Identity Server timing out since the Identity Server was started. This would result in an LDAP Service Not Available error.
Connection Waits Aborted Due To Closed Pool	The number of times that an LDAP connection wait terminated because of a closed connection pool since the Identity Server was started. This would normally be caused by an LDAP replica failing while the user thread is waiting for the connection. This would result in an LDAP Service Not Available error.

14.7 Enabling Identity Server Audit Events

All user and administrator actions can be logged to Novell Audit. You can generate a Novell Audit logging event to indicate whether authentications are successful or unsuccessful. The following steps assume that you have already set up Novell Audit on your network. For more information, see "Enabling Auditing" in the *Novell Access Manager 3.1 SP2 Administration Console Guide*.(To set up auditing, click *Auditing* > *Auditing*.)

1. In the Administration Console, click *Devices* > *Identity Server* > *Servers* > *Edit* > *Logging*.
2. In the *Novell Audit Logging* section, select *Enabled*.
3. Select the events for notification.

   Select All: Select this option for all events. Otherwise, select one or more of the following:

Event	Description
Login Provided	Generated when an identity provider sends authentication to a service provider. Role assignment audit events are included in authentication audit events for the Identity Server.
Login Provided Failure	Generated when an identity provider attempts to send authentication to a service provider but fails.
Login Consumed	Generated when a user is authenticated either locally or by an external identity provider. Role assignment audit events are included in authentication audit events for the Identity Server.
Login Consumed Failure	Generated when the Identity Server initiates authentication, but the process fails.
Logout Provided	Generated when an identity provider sends a logout request to a service provider that it has authenticated.
Logout Local	Generated when the Identity Server receives a logout command from the user.
Federation Request Sent	Generated when a service provider attempts to federate with an identity provider.

Event	Description
Federation Request Handled	Generated by the Identity Server when processing a request for federation.
Defederation Request Sent	Generated by the identity provider when a request for defederation is sent to another provider.
Defederation Request Handled	Generated when the Identity Server processes a request for defederation.
Register Name Request Handled	Generated when the Identity Server processes a request for changing a name identifier.
Attribute Query Request Handled	Generated by the Identity Server when processing an attribute request from a service provider.
Web Service Query Handled	Generated when a Web service query request is sent to an identity provider.
Web Service Modify Handled	Generated when Web service modify request is sent to an identity provider.
User Account Provisioned	Generated by the Identity Server when functioning as an identity consumer and when an account has been provisioned.
User Account Provisioned Failure	Generated by the Identity Server when functioning as an identity consumer and when account provisioning has failed.
LDAP Connection Lost	Generated when the LDAP connection is lost.
LDAP Connection Reestablished	Generated when the LDAP connection is reestablished.
Server Started	Generated when the server gets a start command from the server communications module.
Server Stopped	Generated when the server gets a stop command from the server communications module.
Server Refreshed	Generated when the server gets a refresh command from the server communications module.
Intruder Lockout Detected	Generated when an attempt to log in as a particular user with an invalid password has occurred more times than is allowed by the directory.
Component Log Severe Messages	Logged for all component messages with level of Severe.
Component Log Warning Messages	Logged for all component messages with level of Warning.

4 Click *Apply*, then *OK*.

5 Click *Servers* > *Update Servers*.

Restart the Novell Audit server.

14.8 Monitoring Identity Server Alerts

The Alerts page allows you to view information about current Java alerts and to clear them. An alert is generated whenever the Identity Server detects a condition that prevents it from performing normal system services.

1. In the Administration Console, click *Devices* > *Identity Servers* > *[Name of Server]* > *Alerts* tab.
2. To delete an alert from the list, select the check box for the alert, then click *Acknowledge Alert(s)*. To remove all alerts from the list, click the *Severity* check box, then click *Acknowledge Alert(s*).
3. Click *Close*.
4. (Optional) To verify that the problem has been solved, click *Identity Servers* > *[Name of Server]* > *Health* > *Update from Server*.

14.9 Viewing the Command Status of the Identity Server

Commands are issued to an Identity Server when you make configuration changes and when you select an action such as stopping or starting the Identity Server.

Certain commands, such as start and stop commands, retry up to 10 times before they fail. The first few retries are spaced a few minutes apart, then they move to 10-minute intervals. These commands can take over an hour to result in a failure. As long as the command is in the retry cycle, the command has a status of pending.

- If you do not want to wait for the cycle to complete, you need to manually delete the command.
- If you enter the same command and it succeeds before the first command has completed its retry cycle, the first command always stays in the pending state. You need to manually delete the command.

This section describes the following tasks related to commands:

- Section 14.9.1, “Viewing the Status of Current Commands,” on page 347
- Section 14.9.2, “Viewing Detailed Command Information,” on page 348

14.9.1 Viewing the Status of Current Commands

The Command Status page lists scheduled events and the current status of each event. A new command appears in the list each time you change a configuration. The commands remain listed until you delete them.

1. In the Administration Console, click *Devices* > *Identity Servers*.
2. Click the *Command Status* link for the server.
3. To delete a command, select it and click *Delete*.
4. Click *Refresh* to refresh the display.

The following table describes the columns on the Command Status page:

Column Name	Description
Name	Lists the Identity Server name.
Status	Lists the status of each server.
Type	Displays type of command issued to the server.
Admin	Displays the credentials of the administrator who performed the command.
Date & Time	The date and time that the command was issued. Date and time entries are specified in the local time.

14.9.2 Viewing Detailed Command Information

To view information about an individual command:

1. In Administration Console, click *Devices* > *Identity Servers* > *[Name of Server]* > *Command Status*.
2. Click the name of a command to get detailed information. The following information is displayed:

   Name: The Identity Server name.

   Type: The type of command issued to the server.

   Admin: The distinguished name of the admin user who performed the command.

   Status: The status of the server command.

   Last Executed On: The date and time that the command was executed.
3. To determine if any problems occurred, view the *Command Execution Details* section.

   For a command that fails because the Administration Console cannot communicate with the Identity Server, the page displays the following additional fields:

   Number of Tries: Specifies the number of times the command was executed.

   Command Try Log: Lists each try and the results.
4. Select one of the following actions:
   - **Delete:** To delete a command, click *Delete*. Click *OK* in the confirmation dialog box.
   - **Refresh:** To update the current cache of recently executed commands, click *Refresh*.
5. Click *Close* to return to the Command Status page.

14.10 Tuning the Identity Server for Performance

Use the following information to improve the performance of your Identity Server cluster.

14.10.1 Basic Tuning Options

The following Access Manager components and features can affect the performance of the Identity Server cluster.

LDAP User Stores: This critical component can be a major cause for slowness, depending upon configuration, hardware, and the layout of the directory. Configure search contexts to avoid LDAP searches that traverse the entire tree.

L4 Switches: If the switch is slow or misconfigured, it can severely impact performance. You need to make sure the switch has ample capacity to handle the traffic. If possible, clustered Identity Servers should be plugged directly into the switch or segmented accordingly. It is also critical that you enable sticky bit/persistence on the L4 switch. When this feature is not enabled, the product handles the traffic correctly, but the system can run up to 50% slower than when persistence is enabled. For tips on how to set up the L4 switch, see "Configuration Tips for the L4 Switch " in the *Novell Access Manager 3.1 SP2 Setup Guide*.

Enabled Protocols: On the General Configuration page (click *Devices* > *Identity Servers* > *Edit*), you can select which protocols to enable. The Liberty protocol needs to be enabled, but each additional protocol adds a little processing overhead. Do not enable protocols unless you are using them.

Session Failover: On the Cluster Details page (click *Devices* > *[Name of Cluster]*), you can set up session failover so that if an Identity Server in the cluster goes down, the user does not lose any session data. This feature adds some overhead, because the Identity Servers need to share some authentication information. You need to balance the need to preserve user session data with the increase in authentication traffic. For best performance, you should specify the minimum number of peers.

Limit User Sessions: On the General Configuration page (click *Devices* > *Identity Servers* > *Edit*), you can select to limit the number of sessions a user can have. When a user is limited to a specific number of sessions, the Identity Servers must check with the other servers in the cluster before establishing a new session. This check adds a little bit of overhead to each new authentication request.

Authentication Timeouts: For each contract (click *Devices* > *Identity Servers* > *Edit* > > *Local* > *Contracts* > *[Name of Contract]*), you need to specify an authentication timeout. Short timeouts generate more authentication traffic. Carefully consider the security requirements for your resources and set limits that meet the requirements. If you only need to verify that the users are actively using a session, have all these protected resources use the same contract or have them share the same activity realm.

Logging: You need to manage the size and number of log files as well as the logging level. You should increase the log level to Debug only when you are troubleshooting a problem. As soon as the problem is resolved, you should reduce the log level. You should also have a schedule for checking the number and size of the log files and for removing the older log files.

Auditing: You need to carefully select the events that you audit. Selecting all events that are available for the Access Manager components can impact performance. For example, the Login Provided event generates an event every time a user authenticates. If you have many users, this one event could impact performance. You need to analyze your needs. Are you really interested in who logged in, or are you more interested in who failed to log in?

14.10.2 Disabling User Profile Objects

If you are not using the default configuration for storing Form Fill secrets and you have not enabled persistent federation between identity and service providers, you can disable the creation of objects under the LibertyUserProfile container in the configuration datastore. The default behavior is to create an object in this container for every user accessing the system, and the login process checks for a matching user in this container.

If you have hundreds of thousands of users, the following symptoms might indicate that the user profile objects are slowing down the login process:

- On the Administration Console, the ndsd process (Linux) or the NDS Server (Windows) is running at 100%.
- Running the backup utility is very slow.
- Logging in to the Administration Console is very slow.

To discover whether profile objects might be causing a slowdown, open an LDAP browser (or in the Administration Console, select the *View Objects* task in the menu bar). Expand the following objects: novell > accessManagerContainer > nids > cluster. Expand the SCC objects, and look for objects stored in LibertyUserProfile objects.

- If you have only a few hundred of these objects, user profile objects are not slowing the authentication process.
- If you have thousands of these objects, user profile objects are probably causing a slowdown. You can speed up authentication by disabling the use of these objects. When you do this, the Identity Server no longer creates objects in the LibertyUserProfile container, and it does not try to match an authenticating user with a profile object.

To prevent the creation and use of user profile objects, make the following modifications to your Identity Server configuration:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Liberty* > *Web Service Provider*.
2. Disable the following profiles:
   - Personal Profile
   - Employee Profile
   - Custom Profile
3. Either disable the Credential Profile (which also disables using Form Fill or Identity Injection with credentials) or enable the Credential Profile and modify its default configuration:
   - **3a** Click *Credential Profile*.
   - **3b** Select to store secrets either with the *Extended Schema User References* option or with the *Novell Secret Store User Store References* option.

     When the Credential Profile is enabled, the default behavior is to create user profile objects and store the secrets there. You must configure one of these other options to store the secrets. For more information about these options, see Section 3.1.4, “Configuring a User Store for Secrets,” on page 109.
4. Click *OK* twice, then update the Identity Server.

5 To disable the use of the user profile objects:

5a Log in to the Identity Server machine as the `root` user.

5b Open the `web.xml` file.

Linux: `/var/opt/novell/tomcat5/webapps/nidp/WEB-INF/`

Windows Server 2003: `\Program Files\Novell\Tomcat\webapps\nidp\WEB-INF/`

Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF/`

5c Add the following lines to the file:

```
<context-param>
  <param-name>cpAuthorityType</param-name>
  <param-value>memory</param-value>
</context-param>
```

5d Restart Tomcat.

Linux: Enter the following command:

```
/etc/init.d/novell-tomcat5 restart
```

Windows: Enter the following commands:

```
net stop Tomcat5
net start Tomcat5
```

5e Make this change on each Identity Server in the cluster.

14.10.3 Configuring a Specific IP Address for Proxied Requests

The default behavior for the Identity Server is to use the same IP address for incoming client requests, for proxied requests, and for management tasks. You can improve performance by separating this traffic into separate pools via IP addresses. You can also use the IP addresses to route the traffic so that it remains behind the firewall.

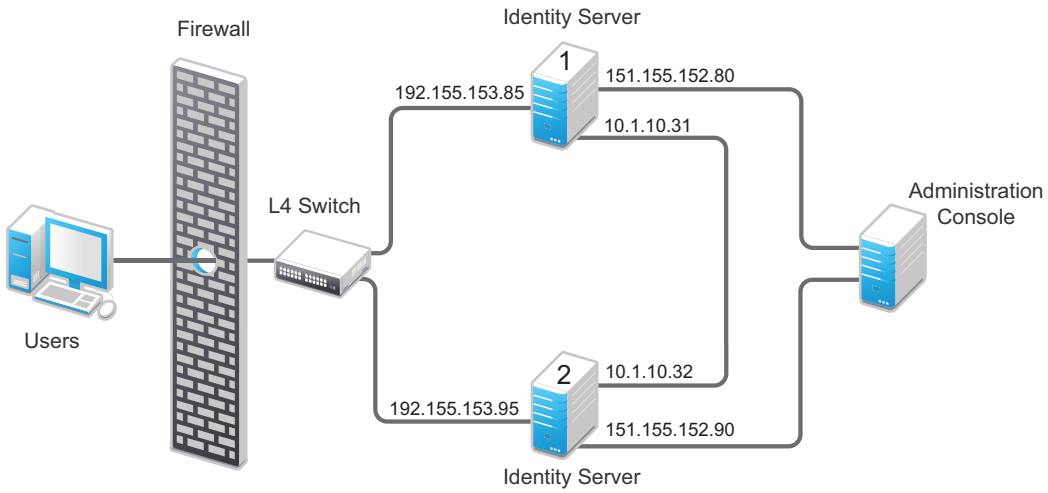
In version 3.1 SP2 IR1 and later, you can specify the IP address that an Identity Server uses for proxied requests to other members of the cluster. A proxied request is sent to another member of a cluster when the request is not sent to the authoritative server.

An authoritative server is the cluster member that holds the authentication information for a given user session. For a request associated with a given session to be processed, it must be routed or proxied to the authoritative cluster member. If an L4 switch sends a request to a non-authoritative cluster member, that cluster member proxies that request to the authoritative cluster member.

You can also specify the IP address for the communication that takes place between the Identity Server and the Administration Console for management tasks. This includes configuration updates, health checks, and statistics. To configure this IP address, log in to the Administration Console, then click *Devices > Identity Servers > [Name of Identity Server]*.

Figure 14-1 illustrates a configuration with a two-member cluster. The L4 switch sends client traffic to the Identity Servers by using the IP addresses that start with 192. The IP addresses that start with 10 are used to route proxied requests to the cluster members. The IP addresses starting with 151 are used for the management traffic with the Administration Console.

Figure 14-1 *Two-Member Identity Server Cluster*



To specify the IP address for the proxied requests on the SOAP channel:

1. Gather the required information. For each Identity Server in the cluster, you need to know the following information:
   - Management IP address. (To get this value or modify the value, click *Devices* > *Identity Servers* > *[Name of Identity Server]*.)
   - IP address or IP address with port that is available to use for proxied requests.
2. Log in to the Identity Server as the `root` user.
3. Change to the `WEB-INF` directory:

   Linux: `/var/opt/novell/tomcat5/webapps/nidp/WEB-INF/`

   Windows Server 2003: `\Program Files \Novell\Tomcat\webapps\nps\WEB-INF/`

   Windows Server 2008: `\Program Files (x86)\Novell\Tomcat\webapps\nps\WEB-INF/`
4. Open the `web.xml` file for editing.
5. Add a proxyAddessMap parameter entry to the file.

```
<context-param>
    <param-name>proxyAddressMap</param-name>
    <param-value>Managament_IP, unused, Proxied_Request_IP
        </param-value>
</context-param>
```

The <param-value> element specifies the IP addresses that are used by the other members of the cluster. It is a comma separated list of IP addresses. You need a value entry for each member of the cluster, except the cluster member you are configuring. A member does not send proxied requests to itself, so you do not need to add it. Each value entry must contain three IP addresses:

- Replace *Management_IP* with the management IP address of the Identity Server. You cannot specify a port with this entry.
- Replace *unused* with just a comma. If you have configured this feature for the Access Gateway, this IP address entry is used for the reverse proxy IP address. The Identity Server does not have a reverse proxy.
- Replace *Proxied_Request_IP* with the address to use for the proxied requests (also called the SOAP back channel). You can specify a port with this entry, such as 151.155.152.90:445.

For Identity Server 1 in Figure 14-1, the entry should look similar to the following lines:

```
<context-param>
   <param-name>proxyAddressMap</param-name>
   <param-value>151.155.152.90,,10.1.10.32</param-value>
</context-param>
```

If your cluster has three or more members, you need to add addresses for the other members. The following example shows an entry for Identity Server 1 in Figure 14-1 if the cluster contained a third member.

```
<context-param>
   <param-name>proxyAddressMap</param-name>
   <param-value>151.155.152.90,,10.1.10.32,
       151.155.152.100,,10.1.10.33</param-value>
</context-param>
```

6 Save the file.

7 Restart Tomcat:

Linux: /etc/init.d/novell-tomcat5 restart

Windows: Enter the following commands:

```
net stop Tomcat5
net start Tomcat5
```

8 Repeat Step 2 through Step 7 for each cluster member, modifying the <param-value> element to contain the addresses for the other members of the cluster.

14.10.4 Configuring Java Memory Allocations

The Tomcat configuration file controls the amount of memory that Tomcat can allocate for Java. If you have installed your Identity Server on a machine with the recommended 4 GB of memory, you can modify two parameters in this file to improve performance under heavy load:

- "Modifying the Java Parameters on Linux" on page 353
- "Modifying the Java Parameters on Windows" on page 354

Modifying the Java Parameters on Linux

1 Log in to the Identity Server as the root user.

2 Open the Tomcat configuration file for editing.

`/var/opt/novell/tomcat5/conf/tomcat5.conf`

3 Find the following line in the file:

```
JAVA_OPTS="-server -Xmx1024m -Xms512m -Xss128k -XX:+UseConcMarkSweepGC"
```

4 Replace the `-Xmx` value (the default is 1024) with 2048.

This allows Java to use 2 GB of memory.

5 Find the following line in the file:

```
JAVA_OPTS="${JAVA_OPTS} -Dnids.freemem.threshold=0"
```

6 Change the `-Dnids.freemem.threshold` value from 0 to a value between 5 and 15.

This prevents user sessions from using up all the memory and ensures that there is free memory available so that the other internal Java processes can continue to function. When this threshold is reached, the user receives a 503 server busy message and a threshold error message is logged to the `catalina.out` file.

7 Save your changes, then restart Tomcat.

8 Copy the modified file to each Identity Server in the cluster, then restart Tomcat on each machine.

Modifying the Java Parameters on Windows

1 Log in to the Identity Server as the administrator.

2 Open the Tomcat configuration utility.

Windows Server 2003: `/Program Files/Novell/Tomcat/bin/tomcat5w.exe`

Windows Server 2008: `/Program Files (x86)/Novell/Tomcat/bin/tomcat5w.exe`

3 Click the *Java* tab.

4 In the *Java options* section, find the following line:

```
-Dnids.freemem.threshold=0
```

If the line does not exist, you need to add it.

5 Change the `-Dnids.freemem.threshold` value from 0 to a value between 5 and 15.

This prevents user sessions from using up all the memory and ensures that there is free memory available so that the other internal Java processes can continue to function. When this threshold is reached, the user receives a 503 server busy message and a threshold error message is logged to the `stdout.log` file.

6 Change the *Maximum memory pool* size to 2048.

This allows Java to use 2 GB of memory.

7 Save your changes, then restart Tomcat.

8 Repeat these steps for each Identity Server in your cluster.

Troubleshooting the Identity Server and Authentication

15

This section discusses the following topics:

Identity Server logging information can be found in Section 14.3, “Configuring Component Logging,” on page 324 and Section 14.4, “Configuring Session-Based Logging,” on page 327.

15.1 Useful Networking Tools for the Linux Identity Server

You can use the following tools (Linux and open source) to troubleshoot network problems:

- **netstat:** Displays information related to open ports on your server. Lets you view listeners and various IP addresses, such as the TCP output state.
- **iptables:** Allows you to change the default ports (8080 and 8443) to the standard ports (80 and 443) for HTTP traffic. See Section 1.5, “Translating the Identity Server Configuration Port,” on page 36.
- **netcat:** A networking utility that reads and writes data across network connections, using the TCP/IP protocol. Netcat is useful for checking connectivity with the user store.
- **ldapsearch:** An LDAP search tool useful for the Administration Console and Identity Server. For example, you can generate an LDAP search/bind matching what the Identity Server sends, to confirm whether an issue is with the Identity Server JAR files.
- **tcpdump:** A command line tool for monitoring network traffic. Captures and displays packet headers and matches them against a set of criteria.
- **LDAP Browser/Editor:** Lets you export configuration information to a file, and to confirm that Access Manager objects and attribute values are valid in an AccessManagerContainer. A number of open source versions are available from the Internet.

15.2 Troubleshooting 100101043 and 100101044 Liberty Metadata Load Errors

The Identity Server is the identity provider for the other Access Manager components. The Access Gateways, ESP-enabled SSL VPN servers, and J2EE Agents have Embedded Service Providers. When a device is imported into the Administration Console and an Identity Server configuration is selected for them, a trusted relationship is established with the Identity Server by using test

certificates. When you change these certificates or change from using HTTP to HTTPS, you need to make sure that the trusted relationship is reestablished. Metadata is used for establishing trusted relationships.

The metadata exchanged between service providers and identity providers contains public key certificates, key descriptors for message signing, a URL for the SSO service, a URL for the SLO (single logout) service, and so on. With Access Manager, this metadata is accessible on both the Identity Server and the Embedded Service Provider of the device. Errors are generated when either the identity provider could not load the service provider's metadata (100101043), or the service provider could not load the metadata of the identity provider (100101044).

If users are receiving either of these errors when they attempt to log in, verify the following:

- Section 15.2.1, "The Metadata," on page 356
- Section 15.2.2, "DNS Name Resolution," on page 357
- Section 15.2.3, "Certificate Names," on page 358
- Section 15.2.4, "Certificates in the Required Trust Stores," on page 359

If these steps do not solve your problem, try the following:

- Section 15.2.5, "Enabling Debug Logging," on page 360
- Section 15.2.6, "Testing Whether the Provider Can Access the Metadata," on page 362
- Section 15.2.7, "Manually Creating Any Auto-Generated Certificates," on page 363
- For information about metadata validation process and the flow of events that occur when accessing a protected resource on the Access Gateway, see "Troubleshooting 100101043 and 100101044 Errors in Access Manager" (http://www.novell.com/coolsolutions/appnote/19456.html).

15.2.1 The Metadata

If you change the base URL of the Identity Provider, all service providers, including Embedded Service Providers, need to be updated so that they use the new metadata:

- "Embedded Service Provider Metadata" on page 356
- "Service Provider Metadata" on page 357

Embedded Service Provider Metadata

If you change the base URL of the Identity Provider, all Access Manager devices that have an Embedded Service Provider need to be updated so that new metadata is imported. To force a re-import of the metadata, you need to configure the device so it doesn't have a trusted relationship with the Identity Server, update the device, reconfigure the device for a trusted relationship, then update the device. The following steps explain how to force the Access Gateway to re-import the metadata of the Identity Server.

1. In the Administration Console, click *Devices* > *Access Gateways* > *Edit* > *Reverse Proxies/Authentication*.
2. Select *None* for the *Identity Server Cluster* option, click *OK* twice, then update the Access Gateway.

3 Click *Edit* > *Reverse Proxies/Authentication*.

4 Select an Identity Server configuration for the *Identity Server Cluster* option, click *OK* twice, then update the Access Gateway.

Service Provider Metadata

If you have set up federation with another provider over the Liberty, SAML 1.1, SAML 2.0, CardSpace, or WS Federation protocol and you change the base URL of the Identity Server, you need to update the provider with the new metadata to reestablish the trusted relationship. If the provider is another Identity Server, follow the procedure below to update the metadata; otherwise, follow the provider's procedures.

1 In the Administration Console of the provider, click *Devices* > *Identity Servers* > *Edit* > *[Protocol]* > *[Provider]* > *Metadata*.

2 Click *Reimport*.

3 Follow the steps in the wizard.

For more information, see Section 7.7, "Managing Metadata," on page 207.

15.2.2 DNS Name Resolution

When the service provider tries to access the metadata on the identity provider, it sends the request to the hostname defined in the base URL configuration of the Identity Server. The base URL in the Identity Server configuration is used to build all the metadata end points.

To view the metadata of the Identity Server with a DNS name of `idpcluster.lab.novell.com`, enter the following URL:

```
https://idpcluster.lab.novell.com:8443/nidp/idff/metadata
```

Scan through the document and notice the multiple references to `https://idpcluster.lab.novell.com/...` You should see lines similar to the following:

```
<md:SoapEndpoint>
   https://idpcluster.lab.novell.com:8443/nidp/idff/soap
</md:SoapEndpoint>

<md:SingleLogoutServiceURL>
   https://idpcluster.lab.novell.com:8443/nidp/idff/slo
</md:SingleLogoutServiceURL>

<md:SingleLogoutServiceReturnURL>
   https://idpcluster.lab.novell.com:8443/nidp/idff/slo_return
</md:SingleLogoutServiceReturnURL>
```

The Embedded Service Provider of the Access Gateway must be able to resolve the `idpcluster.lab.novell.com` hostname of the Identity Server. To test that it is resolvable, send a `ping` command with the hostname of the Identity Server. For example, from the Access Gateway:

```
ping idpcluster.lab.novell.com
```

The same is true for the Identity Server. It must be able to resolve the hostname of the Access Gateway. To discover the URL for the Access Gateway metadata:

1. In the Administration Console, click *Devices* > *Access Gateways* > *Edit* > *Reverse Proxy/ Authentication*.
2. View the *Embedded Service Provider* section.

   The URL of the metadata is displayed in this section.

To view the metadata, enter the displayed URL. Scan through the document and notice the multiple references to the hostname of the Access Gateway. You should see lines similar to the following. In these lines, the hostname is ag1.provo.novell.com.

```
<md:SoapEndpoint>
   http://ag1.provo.novell.com:80/nesp/idff/spsoap
</md:SoapEndpoint>

<md:SingleLogoutServiceURL>
   http://ag1.provo.novell.com:80/nesp/idff/spslo
</md:SingleLogoutServiceURL>

<md:SingleLogoutServiceReturnURL>
   http://ag1.provo.novell.com:80/nesp/idff/spslo_return
</md:SingleLogoutServiceReturnURL>
```

To test that the Identity Server can resolve the hostname of the Access Gateway, send a `ping` command with the hostname of the Access Gateway. For example, from the Identity Server:

```
ping ag1.provo.novell.com
```

To view sample log entries that are logged when a DNS name cannot be resolved, see "The Embedded Service Provider Cannot Resolve the Base URL of the Identity Server" on page 361.

15.2.3 Certificate Names

Make sure the certificates for the Identity Server and the Embedded Service Provider match the hostnames defined in the metadata URL (see Section 15.2.2, "DNS Name Resolution," on page 357).

When the Identity Server and the Access Gateway are enabled for HTTPS, all communication to these devices requires that the devices send back a server certificate. Not only must the certificate be assigned to the appropriate device, but the subject name of the device certificate must match the hostname of the device it is assigned to.

To verify the certificate name of the Identity Server certificate:

1. In the Administration Console, click *Devices* > *Identity Servers* > *Edit*.
2. Click the *SSL Certificate* icon.

   The SSL Connector keystore is displayed

3 Verify that the subject name of the certificate matches the DNS name of the Identity Server.

- If the names match, a certificate name mismatch is not causing your problem.
- If the names do not match, you need to either create a certificate that matches or import one that matches. For information on how to create a certificate for the Identity Server, see "Configuring Secure Communication on the Identity Server" in the *Novell Access Manager 3.1 SP2 Setup Guide*.

To verify the certificate name of the Access Gateway certificate:

1 In the Administration Console, click *Devices* > *Access Gateways* > *Edit* > *[Name of Reverse Proxy]*.

2 Read the alias name of the server certificate, then click the *Server Certificate* icon.

3 Verify that the Subject name of the server certificate matches the published DNS name of the proxy service of the Access Gateway.

- If the names match, a certificate name mismatch is not causing your problem.
- If the names do not match, you need to either create a certificate that matches or import one that matches. For information on how to create an Access Gateways certificate, see "Configuring the Access Gateway for SSL and Other Security Features" in the *Novell Access Manager 3.1 SP2 Access Gateway Guide*.

To view sample log entries that are logged to the `catalina.out` file when the certificate has an invalid name, see "The Server Certificate Has an Invalid Subject Name" on page 362.

15.2.4 Certificates in the Required Trust Stores

Make sure that the issuers of the Identity Server and Embedded Service Provider certificates are added to the appropriate trusted root containers.

When the server certificates are sent from the identity provider to the service provider client, and from the service provider to the identity provider client, the client needs to be able to validate the certificates. Part of the validation process is to confirm that the server certificate has been signed by a trusted source. To do this, the issuers of the server certificate (intermediate and trusted roots) must be imported into the correct trusted root stores:

- The intermediate and trusted roots of the Embedded Service Provider certificate must be imported into the NIDP Trust Store.
- The intermediate and trusted roots of the Identity Server certificate must be imported into the ESP Trust Store.

If you use certificates generated by the Administration Console CA, the trusted root certificate is the same for the Identity Server and the Embedded Service Provider. If you are using external certificates, the trusted root certificate might not be the same, and there might be intermediate certificates that need to be imported.

To verify the trusted root certificates:

1 In the Administration Console, click *Security* > *Certificates*.

2 Determine the issuer of the Identity Server certificate and the Embedded Service Provider certificate:

2a Click the name of the Identity Server certificate, note the name of the Issuer, then click *Close*.

2b Click the name of the Embedded Service Provider certificate of the Access Gateway, note the name of the Issuer, then click *Close*.

2c (Conditional) If you do not know the names of these certificates, see Section 15.2.3, "Certificate Names," on page 358.

3 To verify the trusted root for the Identity Server, click *Devices* > *Identity Servers* > *Edit* > *Security* > *NIDP Trust Store*.

4 In the *Trusted Roots* section, scan for a certificate subject that matches the issuer of the Embedded Service Provider certificate, then click its name.

- If the Issuer has the same name as the Subject name, then this certificate is the root certificate.
- If the Issuer has a different name than the Subject name, the certificate is an intermediate certificate in the chain. Click *Close*, and make sure another certificate in the trust store is the root certificate. If it isn't there, you need to import it and any other intermediate certificates between the one you have and the root certificate.

5 To verify the trusted root for the Embedded Service Provider, click *Devices* > *Access Gateways* > *Edit* > > *Service Provider Certificates* > *Trusted Roots*.

6 In the Trusted Roots section, scan for a certificate subject that matches the issuer of the Identity Server certificate, then click its name.

- If the Issuer has the same name as the Subject name, then this certificate is the root certificate.
- If the Issuer has a different name than the Subject name, the certificate is an intermediate certificate in the chain. Click *Close*, and make sure another certificate in the trust store is the root certificate. If it isn't there, you need to import it and any other intermediate certificates between the one you have and the root certificate.

7 (Optional) If you have clustered your Identity Servers and Access Gateways and you are concerned that not all members of the cluster are using the correct trusted root certificates, you can re-push the certificates to the cluster members.

7a Click *Auditing* > *Troubleshooting* > *Certificates*.

7b Select the Trust Store of your Identity Servers and Access Gateways, then click *Re-push certificates*.

7c Update the Identity Severs and Access Gateways.

7d Check the command status of each device to ensure that the certificate was pushed to the device. From the Identity Servers page or the Access Gateways page, click the *Commands* link.

To view sample log entries that are logged to the `catalina.out` file when a trusted root certificate is missing, see "Trusted Roots Are Not Imported into the Appropriate Trusted Root Containers" on page 362.

15.2.5 Enabling Debug Logging

You can enable Identity Server logging to dump more verbose Liberty information to the `catalina.out` file on both the Identity Server and the Embedded Service Provider of the Access Gateway.

1 In the Administration Console, click *Devices* > *Identity Servers* > *Edit* > *Logging*.

2 Select *Enabled* for *File Logging* and *Echo to Console*.

3 In the *Component File Logger Levels* section, set *Application* and *Liberty* to a *debug* level.

4 Click *OK*, update the Identity Server, then update the Access Gateway.

5 After enabling and applying the changes, duplicate the issue once more to add specific details to the log file for the issue.

If the error is the 100101044 error, look at the log file on the Embedded Service Provider for the error code

If the error is the 100101043 error, look at the log file on the Identity Server for the error code.

On Linux, look at the `catalina.out` file, and on Windows, look at the `stdout.log` file.

6 (Conditional) To view the log files from the Administration Console, click *Auditing* > *General Logging*, then select the file and download it.

7 (Conditional) To view the log files on the device, change to the `log` directory.

- On Linux, change to the `/var/opt/novell/tomcat5/logs` directory.
- On Windows Server 2003, change to the `/Program Files/Novell/Tomcat/logs` directory.
- On Windows Server 2008, change to the `/Program Files (x86)/Novell/Tomcat/logs` directory.

Below are a few typical entries illustrating the most common problems. They are from the `catalina.out` file of the Embedded Service Provider:

- "The Embedded Service Provider Cannot Resolve the Base URL of the Identity Server" on page 361
- "Trusted Roots Are Not Imported into the Appropriate Trusted Root Containers" on page 362
- "The Server Certificate Has an Invalid Subject Name" on page 362

The Embedded Service Provider Cannot Resolve the Base URL of the Identity Server

When the Embedded Service Provider cannot resolve the DNS name of the Identity Server, the metadata cannot be loaded and a hostname error is logged. In the following entries, the Embedded Service Provider cannot resolve the `idpcluster.lab.novell.com` name of the Identity Server.

```
<amLogEntry> 2009-08-06T16:24:56Z INFO NIDS Application: AM#500105024:
AMDEVICEID#esp-09C720981EEE4EB4: AMAUTHID#2CA1168DF7343A42C7879
E707C51A03C: ESP is requesting metadata from IDP https://
idpcluster.lab.novell.com/nidp/idff/metadata </amLogEntry>

<amLogEntry> 2009-08-06T16:24:56Z SEVERE NIDS IDFF: AM#100106001:
AMDEVICEID#esp-09C720981EEE4EB4: Unable to load metadata for Embedded
Service Provider: https://idpcluster.lab.novell.com/nidp/idff/
metadata, error: AM#300101046: AMDEVICEID#esp-09C720981EEE4EB4::   Attempted
to connect to a url with an unresolvable host name
</amLogEntry>

<amLogEntry> 2009-08-06T16:24:56Z INFO NIDS Application: AM#500105039:
AMDEVICEID#esp-09C720981EEE4EB4: AMAUTHID#2CA1168DF7343A42C7879
E707C51A03C: Error on session id 2CA1168DF7343A42C7879E707C51A03C,
error 100101044-esp-09C720981EEE4EB4, Unable to authenticate.
AM#100101044: AMDEVICEID#esp-09C720981EEE4EB4:: Embedded Provider
failed to load Identity Provider metadata </amLogEntry>
```

Trusted Roots Are Not Imported into the Appropriate Trusted Root Containers

When the trusted roots are not imported into the appropriate trusted root containers, a certificate exception is thrown and an untrusted certificate message is logged. In the following log entries, the Embedded Service Provider is requesting metadata from the Identity Server, but the Embedded Service Provider does not trust the Identity Server certificate because the trusted root of the issuer of the Identity Server certificate is not in the Embedded Service Provider's trusted root container.

```
<amLogEntry> 2009-08-05T16:07:53Z INFO NIDS Application: AM#500105024:
AMDEVICEID#esp-09C720981EEE4EB4: AMAUTHID#D983B08C28D35221D13 9D33E5324F98F:
ESP is requesting metadata from IDP https://idpcluster.lab.novell.com/nidp/
idff/metadata </amLogEntry>

<amLogEntry> 2009-08-05T16:07:53Z SEVERE NIDS IDFF: AM#100106001:
AMDEVICEID#esp-09C720981EEE4EB4: Unable to load metadata for Embedded
ServiceProvider: https://idpcluster.lab.novell.com/nidp/idff/metadata, error:
java.security.cert.CertificateException: Untrusted Certificate-  chain </
amLogEntry>

<amLogEntry> 2009-08-05T16:07:53Z INFO NIDS Application: AM#500105039:
AMDEVICEID#esp-09C720981EEE4EB4: AMAUTHID#D983 B08C28D35221D139 D33E5324F98F:
Error on session id D983B08C28D35221D139D33E5324F98F, error 100101044-esp-
09C720981EEE4EB4, Unable to authenticate. AM#100101044: AMDEVICEID#esp-
09C720981EEE4EB4:: Embedded Provider failed to load Identity Provider metadata
</amLogEntry>
```

The Server Certificate Has an Invalid Subject Name

When the certificate has an invalid subject name, the handshake fails. In the log entries below, the Embedded Service Provider is requesting metadata from the Identity Server. The server certificate name does not match, so the Embedded Service Provider is unable to authenticate and get the metadata necessary to establish the trusted relationship.

```
<amLogEntry> 2009-07-05T16:07:53Z INFO NIDS Application: AM#500105024:
AMDEVICEID#esp-09C720981EEE4EB4: AMAUTHID#D983B08C28D35221D139D33 E5324F98F:
ESP is requesting metadata from IDP
https://idpcluster.lab.novell.com/nidp/idff/metadata </amLogEntry>

<amLogEntry> 2009-07-05T16:07:53Z SEVERE NIDS IDFF: AM#100106001:
AMDEVICEID#esp-09C720981EEE4EB4: Unable to load metadata for Embedded Service
Provider: https://idpcluster.lab.novell.com/nidp/idff/metadata, error:
Received fatal alert: handshake_failure </amLogEntry>

<amLogEntry> 2009-07-05T16:07:53Z INFO NIDS Application: AM#500105039:
AMDEVICEID#esp-09C720981EEE4EB4: AMAUTHID#D983B08C28D35221D139D33 E5324F98F:
Error on session id D983B08C28D35221D139D33E5324F98F, error 100101044-esp-
09C720981EEE 4EB4, Unable to authenticate. AM#100101044: AMDEVICEID#esp-
09C720981EEE4EB4: : Embedded Provider failed to load Identity Provider
metadata </amLogEntry>
```

15.2.6 Testing Whether the Provider Can Access the Metadata

To test whether the metadata is available for download, enter the metadata URL of the identity provider and service provider. If the DNS name of the identity provider is `idpcluster.lab.novell.com`, open a browser at the Identity Server and enter the following URL:

```
https://idpcluster.lab.novell.com:8443/nidp/idff/metadata
```

Open a browser on the Access Gateway Service, then enter the same URL.

Because the Access Gateway Appliance does not have a graphical interface, you need to use the `curl` command to test whether the Access Gateway Appliance can access the metadata of the Identity Server. If the DNS name of the identity provider is `idpcluster.lab.novell.com`, enter the following command from the Access Gateway machine:

```
curl -k https://idpcluster.lab.novell.com:8443/nidp/idff/metadata
```

To test whether the Identity Server can access the metadata URL of the Access Gateway, open a browser on the Identity Server machine. If the published DNS name of service provider is `www.aleris.net`, enter the following URL:

```
https://www.aleris.net/nesp/idff/metadata
```

15.2.7 Manually Creating Any Auto-Generated Certificates

Occasionally, there are issues where the subject name was auto-generated and the entire configuration appears to be correct, but the 100101044/100101043 error is still reported. Delete the auto-generated certificate and manually re-create the server certificate, making sure that it is added to the relevant devices and stores.

15.3 Authentication Issues

This section discusses the following issues that occur during authentication:

15.3.1 Authentication Classes and Duplicate Common Names

If users have the same common name and exist in different containers under the same authentication search base, one or more attributes in addition to the common name must be configured for authentication to uniquely identify the user. You can set up an authentication class to handle duplicate common names.

1. Select either the name/password or secure name/password class.
2. Add two properties to the class:
   - **Query:** The value of the Query attribute needs to be a valid LDAP query string. Field names from the JSP login form can be used in the LDAP query string as variables for LDAP attribute values. The variables must be enclosed between two % characters. For example, (&(objectclass=person)(cn=%Ecom_User_ID%)(mail=%Ecom_Email%)) queries for an object of type person that contained a common name equal to the Ecom_User_ID field from the specified JSP form and mail equal to the Ecom_Email field from the same JSP form.

- **JSP:** The JSP property value needs to be the name of a new `.jsp` file that includes all the needed fields for the Query property. The value of this attribute does not include the `.jsp` extension of the file. For example, if you create a new `.jsp` file named `login2.jsp`, the value of the JSP property is `login2`.

  For more information on creating custom login pages that prompt for more than username and password, see Section 2.1, "Customizing the Identity Server Login Page," on page 59.

15.3.2 General Authentication Troubleshooting Tips

- Use LAN traces to check requests, responses, and interpacket delay times.
- In the user store logs, confirm that the request arrived. Check for internal errors.
- If you have created an admin user for the user store, make sure the user has sufficient rights to find the users in the specified the search contexts. For more information about the required rights, see Section 3.1.3, "Configuring an Admin User for the User Store," on page 109.
- Check the user store health and replica layout. See TID 3066352 (http://www.novell.com/support/viewContent.do?externalId=3066352&sliceId=1).
- Ensure that the user exists in the user store and that the user's context is defined as a search context.
- Make sure the Liberty protocol is enabled if you have configured Access Manager devices to use the Identity Server for authentication (click *Identity Servers* > *Edit* > *General Configuration*).
- Check the properties of the class and method. For example, the search format on the properties must match what you've defined on a custom login page. You might be asking for a name/password login, but the method specifies e-mail login criteria.
- Enable authentication logging options (click *Identity Servers* > *Edit* > *Logging*).
- Ensure that the authentication contract matches the base URL scheme. For example, check to see if SSL is used across all components.

15.3.3 Slow Authentication

The following configuration problems can cause slow authentication:

- If authentication is taking up to a minute per user, verify that your DNS server has been enabled for reverse lookups. The JNDI module in the Identity Server sends out a request to resolve the IP address of the LDAP server to a DNS name. If your DNS server is not enabled for reverse lookups, it takes 10 seconds for this request to fail before the Identity Server can continue with the authentication request.
- If your user store resides on SUSE Linux Enterprise Server 10, which installs with a firewall, you must open TCP 524. For more information about the ports that must be open when a firewall separates the user store from other Access Manager components, see "Setting Up Firewalls" in the *Novell Access Manager 3.1 SP2 Setup Guide*.
- If your LDAP user store is large, make sure that the search contexts are as specific as possible to avoid searching the entire tree for a user.

15.3.4 Federation Errors

- Most errors that occur during federation occur because of time synchronization problems between servers. Ensure that all of your servers involved with federation have their time synchronized within one minute.
- When the user denies consent to federate after clicking a Liberty link and logging in at the identity provider, the system displays an error page. The user should acknowledge that federation consent was denied and return to the service provider login page. This is the expected behavior when a user denies consent.

15.3.5 Mutual Authentication Troubleshooting Tips

- LAN traces:
  - Check the SSL handshake and look at trusted root list that was returned.
  - The client certificate issuer must be in the identity provider certificate store and be applied to all the devices in a cluster.
  - Ensure that the user exists and meets the authentication criteria. As the user store administrator, you can search for a subject name (or certificate mapping attributes defined) to locate a matching user.
- Enable the *Show Certificate Errors* option on the Attributes page for the X.509 authentication class. (Click *Identity Servers > Servers > Edit > Local > Classes > [x.509] > Properties*.) Enabling this option provides detailed error messages on the login browser, rather than generic messages.
- Ensure that the certificate subject name matches the user you log in with, if you are chaining methods.
- Use NTRadPing to test installations.
- Verify that the correct UDP port 1812 is specified.
- Verify that the RADIUS server can accept requests from the Identity Server. This might require the NAS-IP-Address attribute along with credentials.
- Verify that the user exists in the user store if multiple methods are added to a contract.
- Verify that user authentication works independent of Access Manager.
- Verify that the NMAS server is local and no tree walks are occurring across the directory.
- Ensure that the NMAS_LOGIN_SEQUENCE property is defined correctly.

15.3.6 Browser Hangs in an Authentication Redirect

If the browser hangs when the user attempts to authenticate at an identity provider, determine whether a new authentication contract was created and set as the default contract on the Identity Server. If this is the case and you have an Access Gateway resource set to accept any contract from the identity provider, you should navigate to the *Overview* tab for the protected resource and specify *Any* again in the *Contract* drop-down menu. Then click *OK*, then update the Access Gateway.

15.3.7 Duplicate Set-Cookie Headers

The Access Manager releases previous to Access Manager 3.1 SP1 allowed a colon in the Set-Cookie header. Because the cookie specifications stipulate that a colon character cannot be used in a cookie, the Set-Cookie header in Access Manager 3.1 SP1 removes the colon and sets a value similar to the following:

```
UrnNovellNidpClusterMemberId=~03~0Bslo~0A~0B~14mop~0C~09; Path=/nidp
```

A second Set-Cookie header is included with the colon value to allow for backward compatibility with devices that have not been upgraded to Access Manager 3.1 SP1. The devices requiring this old style cookie include Identity Servers that haven't been upgraded and any device with an Embedded Service Provider that hasn't been upgraded. The old Set-Cookie header value looks similar to the following:

```
urn:novell:nidp:cluster:member:id=~03~0Bslo~0A~0B~14mop~0C~09; Path=/nidp
```

Both cookies contain the same information. Setting the two cookies does not impact functionality or performance.

15.4 Problems Reading Keystores after Identity Server Re-installation

This can occur if you replace a hard drive and incorrectly reinstall the Identity Server. See "Reinstalling an Identity Server to a New Hard Drive" in the *Novell Access Manager 3.1 SP2 Installation Guide* for the correct procedure.

15.5 After Setting Up the User Store to Use SecretStore, Users Report 500 Errors

If your eDirectory user store is running on SLES 11 64-bit operating system on x86-64 hardware, you can install the NMAS SAML method for SecretStore from the Administration Console, but the eDirectory server is missing the required support libraries.

When users try to enter values for SecretStore entries in a form, they receive the following message:

```
Status: 500 Internal Server Error, Description: Datastore Error
```

To correct the problem, you need to install the missing libraries on your eDirectory server. For instructions, see TID 7006437 (http://www.novell.com/support/viewContent.do?externalId=7006437&sliceId=1).

About Liberty

A

The Liberty Alliance is a consortium of business leaders with a vision to enable a networked world in which individuals and businesses can more easily conduct transactions while protecting the privacy and security of vital identity information.

To accomplish its vision, the Liberty Alliance established an open standard for federated network identity through open technical specifications. In essence, this open standard is a structured version of the Security Assertions Markup Language, commonly referred to as SAML, with the goal of accelerating the deployment of standards-based single sign-on technology.

For general information about the Liberty Alliance, visit the Liberty Alliance Project Web site (http://www.projectliberty.org/).

Liberty resources, including specifications, white papers, FAQs, and presentations, can be found at the Liberty Alliance Resources Web site (http://www.projectliberty.org/liberty/resource_center/).

The following table provides links to specific Liberty Alliance specifications:

Table A-1 *Liberty Alliance Links*

Liberty Specification	Location
Liberty Alliance Project Overview	Liberty Alliance Project Overview (http://www.projectliberty.org/)
Liberty White Papers	Papers (http://www.projectliberty.org/liberty/resource_center/papers)
Identity Federation Specifications	Liberty ID-FF 1.2 Specification (http://www.projectliberty.org/resources/specifications.php#box1)
Web Service Framework Specifications	Liberty ID-WSF 1.1 Specifications (http://www.projectliberty.org/resources/specifications.php#box2a)
Liberty Profile Service Specifications	Liberty Alliance ID-SIS 1.0 Specifications (http://www.projectliberty.org/resources/specifications.php#box3)
OASIS Standards (SAML)	Oasis Standards (http://www.oasis-open.org/specs/index.php#samlv2.0)

Understanding How Access Manager Uses SAML

B

Security Assertions Markup Language (SAML) is an XML-based framework for communicating security assertions (user authentication, entitlement, and attribute information) between trusted identity providers and trusted service providers. For example, an airline company can make assertions to authenticate a user to a partner company or another enterprise application, such as a car rental company or hotel.

The Identity Server allows SAML assertions to be exchanged with trusted service providers that are using SAML servers. Using SAML assertions in each Access Manager component protects confidential information by removing the need to pass user credentials between the components to handle session management.

An identity provider using the SAML protocol generates and receives assertions for authentication, according to the SAML 1.0, 1.1, and 2.0 specifications described on the Oasis Standards Web site (http://www.oasis-open.org/specs/index.php).

This section describes how Access Manager uses SAML. It includes the following topics:

B.1 Attribute Mapping with Liberty

Attribute-based authorization involves one Web site communicating identity information about a subject to another Web site in support of some transaction. However, the identity information might be some characteristic of the subject, such as a role. The attribute-based authorization is important when the subject’s identity is either not important, should not be shared, or is insufficient on its own.

In order to interoperate with trusted service providers through the SAML protocol, the Identity Server distinguishes between different attributes from different SAML implementations. All of the SAML administration is done with Liberty attributes. When you specify which attributes to include in an assertion, or which attributes to use when locating the user from an assertion, these attributes should always be specified in the Liberty format.

In an attribute map, you convert SAML attributes from each vendor’s implementation to Liberty attributes. (See Section 6.1, “Configuring Attribute Sets,” on page 177.)

You can find detailed information about SAML 2.0 on the OASIS Standards Web site (http://www.oasis-open.org/specs/).

B.2 Trusted Provider Reference Metadata

Metadata is generated by the Identity Server and is used for server communication and identification. Metadata can be obtained via URL or XML document, then entered in the system when you create the reference. Metadata is traded with federation partners and supplies various information regarding contact and organization information located at the Identity Server. Metadata is generated automatically for SAML 2.0. You enter it manually for SAML 1.1. (See Chapter 7, "Configuring SAML and Liberty Trusted Providers," on page 187.)

IMPORTANT: The SAML 2.0 and Liberty 1.2 protocols define a logout mechanism whereby the service provider sends a logout command to the trusted identity provider when a user logs out at a service provider. SAML 1.1 does not provide such a mechanism. For this reason, when a logout occurs at the SAML 1.1 service provider, no logout occurs at the trusted identity provider. A valid session is still running at the identity provider, and no credentials need to be entered. In order to log out at both providers, users must navigate to the identity provider that authenticated them to the SAML 1.1 service provider and log out manually.

B.3 Identity Federation

Identity federation is the association of accounts between an identity provider and a service provider, while maintaining privacy protection. From an administrative perspective, this type of sharing can help reduce identity management costs because multiple organizations do not need to independently collect and maintain identity-related data, such as passwords. From the end user's perspective, this results in an enhanced experience by requiring fewer sign-ons.

B.4 Authorization Services

When a user has authenticated to a site or application, the user has access to a resource controlled by a Policy Enforcement Point (PEP). The PEP checks for user access to the desired resource. The user is either granted or denied access to the resource. SAML is used as the communication mechanism between the PEP and a Policy Decision Point (PDP). In Novell product terminology, a PEP could be thought of as the Novell Access Gateway, and the PDP as the Novell Identity Server.

B.5 What's New in SAML 2.0?

SAML 2.0 provides several new features:

- **Pseudonyms:** An arbitrary name assigned by the identity provider to identify a user to a service provider. The identifier has meaning only in the context of the relationship between the relying parties. They can be a principal's e-mail or account name. Pseudonyms are a key privacy feature that inhibits collusion between multiple providers.
- **Metadata:** The SAML metadata specification defines how to express configuration and trust-related data to simplify SAML deployment. Metadata identifies the Identity Servers involved in performing single sign-on between trusted identity providers and service providers.

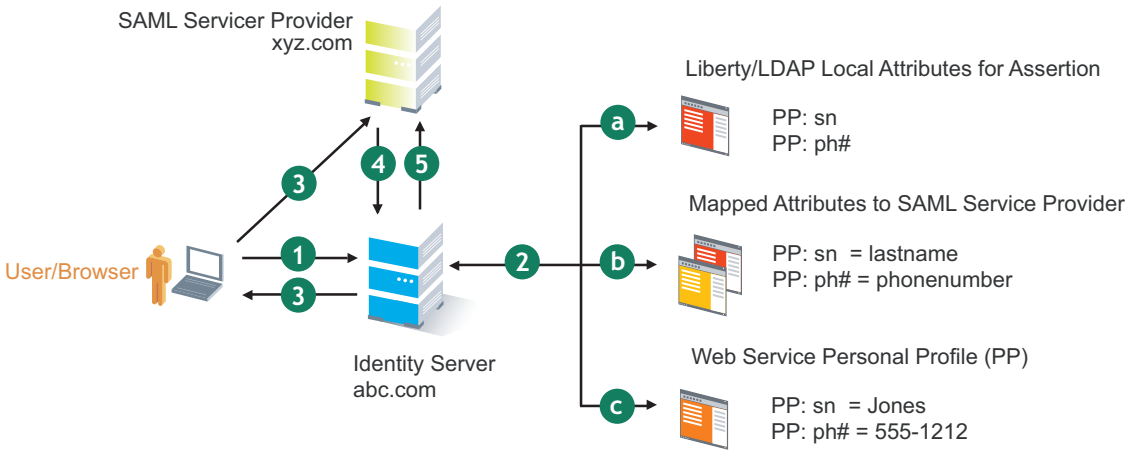
  Metadata includes supported roles, identifiers, supported profiles, URLs, certificates, and keys. System entities must agree upon the data.
- **Encryption:** SAML permits attribute statements, name identifiers, or entire assertions to be encrypted. Encryption ensures that end-to-end confidentiality of these elements can be supported as needed.

- **Attribute profiles:** Profiles simplify how you configure and deploy systems that exchange attribute data. They include:
  - **Basic attribute profile:** Supports string attribute names and attribute values drawn from XML schema primitive type definitions.
  - **X.500/LDAP:** Supports canonical X.500/LDAP attribute names and values.
  - **UUID attribute profile:** Supports using UUIDs as attribute names.
  - **XACML attribute profile:** Defines formats suitable for processing by XACML (Extensible Access Control Markup Language).

B.6 Identity Provider Process Flow

The following illustration provides an example of an Identity Server automatically creating an authenticated session for the user at a trusted SAML service provider. PP indicates a Personal Profile Service as defined by the Liberty specification.

Figure B-1 *SAML Service Provider Process Flow*



1. A user is logged in to the Identity Server at abc.com (the user's identity provider) and clicks a link to xyz.com, a trusted SAML service provider.

   The Identity Server at abc.com generates the artifact. This starts the process of generating and sending the SAML assertion. The HREF would look similar to the following:

   ```
   http://nidp.com/saml/genafct?TARGET=http://xyz.com/index.html&AID=XYZ
   ```

2. The Identity Server processes attributes as follows:

   a. The server looks up LDAP or Liberty-LDAP mapped attributes. (See Section 13.6, "Mapping LDAP and Liberty Attributes," on page 308.) In this example, you use Liberty attributes such as *PP: sn* instead of *surname*. *PP: sn* and *PP: ph#* are attributes that you are sending to xyz.com.

   b. The Identity Server processes these attributes with a SAML implementation-specific attribute.

      Because the identity provider must interoperate with other SAML service providers that probably do not use consistent attribute names, you can map the service provider attributes to your Liberty and LDAP attributes on the Identity Server. In this example, the service

provider names for the Liberty *PP: sn* and *PP: ph#* attributes are *lastname* and *phonenumber*, respectively. (See Section 7.6.1, “Configuring the Attributes Obtained at Authentication,” on page 204.)

c. The Identity Server uses the PP service to look up the values for the user’s *PP: sn* and *PP: ph#* attributes.

   The Identity Server recognizes that the values for the user’s *PP: sn* and *PP: ph#* attributes are *Jones* and *555-1212*, respectively.

3. The Identity Server sends an HTTP redirect with an artifact.

   The Identity Server now has the information to generate a SAML assertion. The Identity Server sends an HTTP redirect containing the artifact back to the browser. The redirect looks similar to the following:

   ```
   http://xyz.com/auth/afct?TARGET=http://xyz.com/index.html&SAMLArtifact
   =<<artifact>>
   ```

4. The remote SAML server requests the assertion.

   The HTTP redirect results in the browser sending the artifact to the SAML server at xyz.com. The SAML server at xyz.com requests the SAML assertion from the Identity Server.

5. The Identity Server sends the assertion to the remote SAML server.

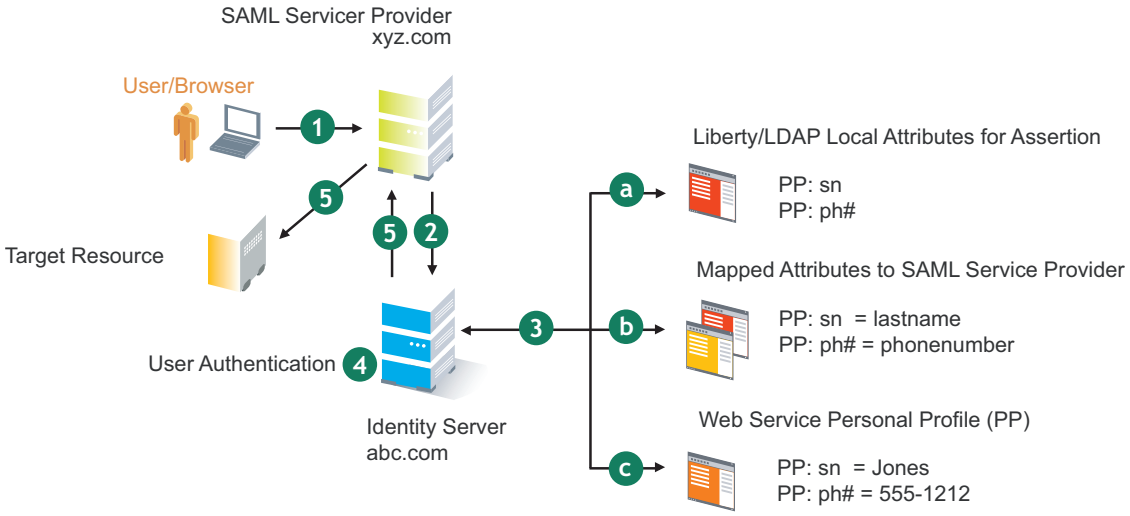
   The remote SAML server receives the artifact and looks up the assertion.The assertion is sent to the SAML server at *xyz.com* in a SOAP envelope. The assertion contains the attributes *lastname=Jones* and *phonenumber=555-1212*.

   The user now has an authenticated session at xyz.com. The xyz.com SAML server redirects the user’s browser to http://xyz.com/index.html, which was referenced in the original HREF in Step 1.

B.7 SAML Service Provider Process Flow

The following illustration provides an example of the authentication process on the consumer side, when a user clicks a link at the SAML service provider (xyz.com) in order to begin an authentication session with an identity provider (such as abc.com). PP indicates a Personal Profile Service as defined by the Liberty specification.

Figure B-2 *SAML Consumer Process Flow*



1. The user clicks a link at xyz.com.

   This generates a SAML assertion intended for the Identity Server at abc.com, which is the identity provider in an Access Manager configuration. After the SAML server generates the artifact, it sends the browser a redirect containing the artifact. The browser is redirected to the identity provider, which receives the artifact. The URL sent to the Identity Server would look similar to the following:

   ```
   http://nidp.com/auth/afct?TARGET=http://abc.com/index.html&SAMLArtifact
   =<<artifact>>
   ```

2. The Identity Server at abc.com receives the assertion.

   The assertion is sent to the Identity Server packaged in a SOAP envelope. In this example, the assertion contains the attributes *lastname=Jones*, and *phonenumber=555-1212*.

3. The Identity Server determines which attributes to use when locating the user.

   The Identity Server must determine how to locate the user in the directory. When you created the SAML service provider reference for xyz.com, you specified which Liberty attributes should be used for this purpose. In this case, the you specified that *PP: sn* and *PP: ph#* should be used.

   a. The Identity Server processes the Liberty attribute map (see Section 13.6, “Mapping LDAP and Liberty Attributes,” on page 308) to the SAML implementation-specific attributes (see Section 7.6.1, “Configuring the Attributes Obtained at Authentication,” on page 204).

      Because this SAML implementation must interoperate with other SAML implementations that probably do not use consistent attribute names, you can map the attributes used by each third-party SAML implementation to Liberty attributes on the Identity Server.

b. The Identity Server receives implementation-specific SAML attribute names.

   The trusted service provider's names for the Liberty *PP: sn* and *PP: ph#* attributes are returned. Using the attribute map, the Identity Server knows that the service provider's names for these attributes are *lastname* and *phonenumber*, respectively.

c. The Identity Server uses the PP service to lookup the values for the user's *PP: sn* and *PP: ph#* attributes.

   The Identity Server now recognizes that the values for the user's *PP: sn* and *PP: ph#* attributes are *Jones* and *555-1212*, respectively. The user's DN is returned to the Identity Server, and the user is authenticated.

4. The user's DN is returned to the Identity Server, and the user is authenticated.
5. The user is redirected to the target resource at xyz.com.

Data Model Extension XML

C

The data model for some Web services is extensible. You can enter XML definitions of data model extensions in a custom profile (for more information, see Section 13.2.1, “Modifying Service and Profile Details for Employee, Custom, and Personal Profiles,” on page 297). Data model extensions hook into the existing Web service data model at predefined locations.

All schema model extensions reside inside of a schema model extension group. The group exists to bind model data items together under a single localized group name and description. Schema model extension groups can reside inside of a schema model extension root or inside of a schema model extension. There can only be one group per root or extension. Each root is hooked into the existing Web service data model. Multiple roots can be hooked into the same location in the existing Web service data model. This conceptual model applies to the structure of the XML that is required to define data model extensions.

The high-level view of the data model extension XML is as follows:

```
<SchemaExtensions>
   <Root>
     <Group>
       <Extension>
          <Group>
           <Extension>...</Extension>
           <Extension>...</Extension>
           ...
          </Group>
       </Extension>
       <Extension>
         <ValueSet>
            <Value/>
            <Value/>
         </ValueSet>
       </Extension>
      ...
    </Group>
  <Root>
<Root>...</Root>
...
</SchemaExtensions>
```

C.1 Elements

The definition of the attributes for each data model extension XML element are as follows:

- “Root Element” on page 376
- “Group Element” on page 376
- “Extension Element” on page 377
- “ValueSet Element” on page 378
- “Value Element” on page 378

Root Element

parent: The unique identifier of the “hook point” in the Web service’s data model. These hook points are defined by the Web service data model schema. These unique identifiers represent the xpaths of each data item within the model schema. Possible values for the parent attribute are listed in Table C-1:

Table C-1 *Root Element*

Personal Profile	/pp:PP/pp:Extension /pp:PP/pp:CommonName/pp:Extension /pp:PP/pp:CommonName/pp:AnalyzedName/pp:Extension /pp:PP/pp:LegalIdentity/pp:Extension /pp:PP/pp:LegalIdentity/pp:VAT/pp:Extension /pp:PP/pp:LegalIdentity/pp:AltID/pp:Extension /pp:PP/pp:EmploymentIdentity/pp:Extension /pp:PP/pp:AddressCard/pp:Extension /pp:PP/pp:AddressCard/pp:Address/pp:Extension /pp:PP/pp:MsgContact/pp:Extension /pp:PP/pp:Facade/pp:Extension /pp:PP/pp:Demographics/pp:Extension
Employee Profile	/ep:EP/ep:Extension /ep:EP/ep:CorpCommonName/ep:Extension /ep:EP/epCorpLegalIdentity/ep:Extension /ep:EP/ep:CorpLegalIdentity/ep:VAT/ep:Extension /ep:EP/ep:CorpLegalIdentity/ep:AltID/ep:Extension
Open Profile	/op:OP/op:Extension /op:OP/op:CustomizableStringsop:Extension

package (required): The Java package name where all classes for this root are implemented. This includes resource description classes and data model instance classes. For example, com.novell.nids.profile.model.extensions.

resourceClass (required): The Java class name of the resource description class that is used to load all resources associated with this root. Because resource description class files are assumed to reside in the root’s package, only the filename is needed. Resource description classes are Java classes that must be created by the person extending the model. You must also extend the com.novell.nidp.resource.NIDPResDesc class.

Group Element

resourceID: The resource ID of the display name of the group. This resource ID is assumed to be a key in the resource bundle supplied by the resource description class file associated with the containing root.

descriptionResourceID: The resource ID of the description of the group. This resource ID is assumed to be a key in the resource bundle supplied by the resource description class file associated with the containing root.

Extension Element

name (required): The name of the data model extension. This name must be the name of the XML element that will be used in the data model.

class (optional): The Java class name of the data model instance class. Because data model instance class files are assumed to reside in the root's package, only the filename is needed. If this attribute is omitted, then the value of the name attribute must be the instance class filename.

syntax: The syntax of this data model extension. Possible values are:

- String
- LocalizedString
- Container

format: Required if the syntax is *String* or *LocalizedString*. The syntax of this data model extension. Possible values are:

- Caselgnore
- CaseExtract
- URI
- URL
- Date
- DateNoYear
- CountryCode
- LanguageCode
- KeyInfo
- Number

upper: The upper bound of a numeric value. Use this attribute only if the format attribute value is Number. The value is a signed integer. If this attribute is omitted, the default value is java.lang.Integer.MAX_VALUE.

lower (optional): The lower bound of a numeric value. This attribute is only used if the format attribute value is Number. The value is a signed integer. If this attribute is omitted, the default value is java.lang.Integer.MIN_VALUE.

min (required): The cardinality of the XML element represented by this data model extension. It is the minimum number of elements allowed. The value is an unsigned integer. If this attribute is omitted, the default value is 0.

max (required): The cardinality of the XML element represented by this data model extension. It is the maximum number of elements allowed. The value is an unsigned integer. If this attribute is omitted, the default value is 1. The value UNBOUNDED may be used to indicate that there are no bounds.

namingClass: (required if syntax equals Container and max is UNBOUNDED). The class that is used as the naming attribute for the container. The class must represent one of the immediate children of the container. This class is used to name each instance of the container.

ValueSet Element

A ValueSet element contains a set of fixed values that a data model entry can contain. If a data model extension has a ValueSet, the user interface to edit the value of that extension limits the user to these values. The ValueSet element has no attributes.

Value Element

A Value element represents a value in a ValueSet. It contains the actual value to be stored in the data model entry and the display name resource ID associated with the value.

resourceID (required): The resource ID of the display name of the value. This resource ID is assumed to be a key in the resource bundle supplied by the resource description class file associated with the containing root.

value (required): The value stored in the data model entry.

name (required): The name of the data model extension. This name must be the name of the XML element that is used in the data model.

C.2 Writing Data Model Extension XML

Data model extension XML must be defined in the namespace novell:liberty:wsf:config:1:0:0 and that namespace must be defined on the SchemaExtensions element. Normally, the namespace prefix wsfc is used. An example of data model extension XML is:

```
<wsfc:SchemaExtensions xmlns:wsfc="novell:liberty:wsf:config:1:0:0">
  <wsfc:Root parent="/pp:PP/pp:Facade/pp:Extension"
     package="com.novell.nidp.liberty.wsf.idsis.ppservice.extensions"
     resourceClass="PPExtensionsResDesc">
   <wsfc:Group resourceId="PP.EXT.FC.GROUP"
      descriptionResourceId="PP.EXT.FC.GROUP.DESC">
   <wsfc:Extension name="AliasName"
       class="FacadeAliasName"
       syntax="String"
       format="CaseIgnore"
       resourceId="PP.EXT.FC.AliasName"
       min="0" max="1"/>
   <wsfc:Extension name="FavoriteURLs"
       class="FacadeFavoriteURLs"
       syntax="String"
       format="CaseExact"
       resourceId="PP.EXT.FC.FavoriteURLs" min="0" max="UNBOUNDED"/>
   </wsfc:Group> </wsfc:Root>
   <wsfc:Root parent="/pp:PP/pp:Demographics/pp:Extension"
      package="com.novell.nidp.liberty.wsf.idsis.ppservice.extensions"
      resourceClass="PPExtensionsResDesc">
   <wsfc:Group resourceId="PP.EXT.DM.GROUP"
      descriptionResourceId="PP.EXT.DM.GROUP.DESC">
   <wsfc:Extension name="EyeColor"
      class="DemographicsEyeColor"
      syntax="String" format="URI"
      resourceId="PP.EXT.DM.EyeColor"
      min="0"
      max="UNBOUNDED">
   <wsfc:ValueSet>
<wsfc:Value resourceId="PP.EXT.DM.HC.Blue" value="urn:pp:dm:blue"/>
```

```
<wsfc:Value resourceId="PP.EXT.DM.HC.Brown" value="urn:pp:dm:brown"/>
<wsfc:Value resourceId="PP.EXT.DM.HC.Green" value="urn:pp:dm:green"/>
<wsfc:Value resourceId="PP.EXT.DM.HC.Gray" value="urn:pp:dm:gray"/>
<wsfc:Value resourceId="PP.EXT.DM.HC.Hazel" value="urn:pp:dm:hazel"/>
</wsfc:ValueSet>
</wsfc:Extension>
</wsfc:Group>
</wsfc:Root>
<wsfc:Root parent="/pp:PP/pp:Extension"
   package="com.novell.nidp.liberty.wsf.idsis.ppservice.extensions"
   resourceClass="PPExtensionsResDesc">
<wsfc:Group resourceId="PP.EXT.AU.GROUP"
  descriptionResourceId="PP.EXT.AU.GROUP.DESC">
<wsfc:Extension name="Automobile"
  class="Automobile"
  syntax="Container"
  resourceId="PP.EXT.Automobile"
  min="0"
  max="UNBOUNDED"
  namingClass="AutomobileLicensePlate">
<wsfc:Group resourceId="PP.EXT.AU.DETAILS.GROUP"
  descriptionResourceId="PP.EXT.AU.DETAILS.GROUP.DESC">
<wsfc:Extension name="AutomobileModel"
  class="AutomobileModel"
  syntax="String"
  resourceId="PP.EXT.AU.Model"
  min="0"
  max="1"/>
<wsfc:Extension name="AutomobileMake"
  class="AutomobileMake"
  syntax="String"
  format="CaseIgnore"
  resourceId="PP.EXT.AU.Make"
  min="0"
  max="1"/>
<wsfc:Extension name="AutomobileLicensePlate"
  class="AutomobileLicensePlate"
  syntax="String"
  format="CaseIgnore"
  resourceId="PP.EXT.AU.LicensePlate"
  min="0" max="1"/>
</wsfc:Group>
</wsfc:Extension>
</wsfc:Group>
</wsfc:Root>
</wsfc:SchemaExtensions>
```