



# Identity Server Guide

## Access Manager 4.0

June 2014

## Legal Notice

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

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

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

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

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

**© 2014 NetIQ Corporation. All Rights Reserved.**

For information about NetIQ trademarks, see <https://www.netiq.com/company/legal/>.

---

# Contents

<b>About NetIQ Corporation</b>	<b>13</b>
<b>About this Book and the Library</b>	<b>15</b>
<b>1 Configuring an Identity Server</b>	<b>17</b>
1.1 Managing a Cluster Configuration	17
1.1.1 Creating a Cluster Configuration	18
1.1.2 Assigning an Identity Server to a Cluster Configuration	22
1.1.3 Configuring a Cluster with Multiple Identity Servers	23
1.1.4 Configuring Session Failover	24
1.1.5 Editing Cluster Details	25
1.1.6 Removing a Server from a Cluster Configuration	26
1.1.7 Enabling and Disabling Protocols	27
1.1.8 Modifying the Base URL	27
1.2 Enabling Role-Based Access Control	28
1.3 Enabling External Attributes Policy	28
1.4 Configuring Secure Communication on the Identity Server	29
1.4.1 Configuring Enhanced Security for Service Provider Communications	30
1.4.2 Viewing the Services That Use the Signing Key Pair	30
1.4.3 Viewing Services That Use the Encryption Key Pair	31
1.4.4 Managing the Keys, Certificates, and Trust Stores	31
1.5 Security Considerations	34
1.5.1 Federation Options	34
1.5.2 Authentication Contracts	34
1.5.3 Forcing 128-Bit Encryption	35
1.5.4 Securing the Identity Server Cookie	36
1.5.5 Configuring the Encryption Method for the SAML Assertion	36
1.5.6 Configuring SAML 2.0 to Sign Messages	37
1.5.7 Blocking Access to Identity Server Pages	39
1.6 Translating the Identity Server Configuration Port	39
1.6.1 Changing the Port on a Windows Identity Server	39
1.6.2 Changing the Port on a Linux Identity Server	40
1.7 Using netHSM for the Signing Key Pair	44
1.7.1 Understanding How Access Manager Uses Signing and Interacts with the netHSM Server	44
1.7.2 Configuring the Identity Server for netHSM	46
<b>2 Customizing Login Pages, Logout Pages, and Messages</b>	<b>61</b>
2.1 Customizing the Identity Server Login Page	61
2.1.1 Selecting the Login Page and Modifying It	62
2.1.2 Configuring the Identity Server to Use Custom Login Pages	73
2.1.3 Troubleshooting Tips for Custom Login Pages	79
2.2 Customizing the Identity Server Logout	80
2.2.1 Rebranding the Logout Page	80
2.2.2 Replacing the Logout Page with a Custom Page	80
2.2.3 Configuring for Local Rather Than Global Logout	81
2.3 Customizing Identity Server Messages	82
2.3.1 Customizing Messages	82
2.3.2 Customizing the Branding of the Error Page	84
2.3.3 Customizing Tooltip Text for Authentication Contracts	85
2.4 Sample Custom Login Pages	86

2.4.1	Modified login.jsp File for Credential Prompts	86
2.4.2	Custom nidp.jsp File with Custom Credentials	89
2.4.3	Custom 3.1 login.jsp File	95
2.4.4	Custom 3.0 login.jsp File	98
2.5	Preventing Cross-site Scripting Attacks	101
2.5.1	Option 1: HTML Escaping	102
2.5.2	Option 2: Filtering	103

### **3 Configuring Local Authentication 105**

3.1	Configuring Identity User Stores	106
3.1.1	Using More Than One LDAP User Store	106
3.1.2	Configuring the User Store	107
3.1.3	Configuring an Admin User for the User Store	111
3.1.4	Configuring a User Store for Secrets	111
3.2	Creating Authentication Classes	120
3.2.1	Creating Basic or Form-Based Authentication Classes	122
3.2.2	Specifying Common Class Properties	123
3.3	Configuring Authentication Methods	125
3.4	Configuring Authentication Contracts	128
3.4.1	Using a Password Expiration Service	132
3.4.2	Using Login Redirect URL Parameters	135
3.4.3	Admin can configure some more parameters that wiUsing Activity Realms	135
3.5	Specifying Authentication Defaults	137
3.5.1	Specifying Authentication Types	138
3.5.2	Creating a Contract for a Specific Authentication Type	138
3.6	Managing Direct Access to the Identity Server	139
3.6.1	Logging In to User Portal	139
3.6.2	Specifying a Target	140
3.6.3	Blocking Access to the User Portal Page	140
3.6.4	Blocking Access to the WSDL Services Page	142

### **4 Configuring Advanced Local Authentication Procedures 145**

4.1	Configuring Social Authentication	145
4.1.1	Use Case	145
4.1.2	Prerequisite	146
4.1.3	Configuring SocialAuthClass	146
4.1.4	Adding Images for Social Authentication Providers	148
4.1.5	Changing the Social Authentication Icons	149
4.1.6	Configuring the Supported Social Authentication Providers for API Keys and API Secrets	149
4.2	Configuring for Two-Factor Authentication Using Time-Based One-Time Password (TOTP)	150
4.2.1	Why Two-Factor Authentication?	150
4.2.2	Prerequisite	150
4.2.3	Configuring TOTP Class, Method, and Contract	150
4.2.4	Registering with Google Authenticator	152
4.2.5	Verifying TOTP Configuration	152
4.3	Configuring Persistent Authentication	153
4.3.1	Frequent Re-authentication Using Password	153
4.3.2	PersistentAuthClass Properties	153
4.3.3	Configuring Persistent Authenticator Class	154
4.3.4	Logging Out of the Persistent Sessions	154
4.3.5	Limitations	155
4.4	Configuring for RADIUS Authentication	155
4.5	Configuring Client Integrity Check	156
4.6	Configuring Mutual SSL (X.509) Authentication	157
4.6.1	Configuring Attribute Mappings	159

4.6.2	Setting Up Mutual SSL Authentication . . . . .	161
4.6.3	Configuring X.509 Authentication to Provide Access Manager Error Message . . . . .	162
4.7	Creating an ORed Credential Class . . . . .	164
4.8	Configuring for OpenID Authentication . . . . .	165
4.9	Configuring Password Retrieval . . . . .	166
4.10	Configuring Access Manager for NESCM . . . . .	169
4.10.1	Prerequisites . . . . .	169
4.10.2	Creating a User Store . . . . .	169
4.10.3	Creating a Contract for the Smart Card . . . . .	171
4.10.4	Assigning the NESCM Contract to a Protected Resource . . . . .	175
4.10.5	Verifying the User's Experience . . . . .	175
4.10.6	Troubleshooting . . . . .	176
<b>5</b>	<b>Configuring for Kerberos Authentication</b>	<b>177</b>
5.1	Prerequisites . . . . .	178
5.2	Configuring Active Directory . . . . .	178
5.2.1	Installing the spn and the ktpass Utilities for Windows Server 2003 . . . . .	179
5.2.2	Creating and Configuring the User Account for the Identity Server . . . . .	179
5.2.3	Configuring the Keytab File . . . . .	180
5.2.4	Adding the Identity Server to the Forward Lookup Zone . . . . .	180
5.3	Configuring the Identity Server . . . . .	181
5.3.1	Enabling Logging for Kerberos Transactions . . . . .	181
5.3.2	Configuring the Identity Server for Active Directory . . . . .	181
5.3.3	Creating the Authentication Class, Method, and Contract . . . . .	182
5.3.4	Creating the bcsLogin Configuration File . . . . .	185
5.3.5	Verifying the Kerberos Configuration . . . . .	186
5.3.6	(Optional) Using the Name/Password Form Authentication . . . . .	186
5.3.7	(Optional) Configuring the Fall Back Authentication Class . . . . .	187
5.4	Configuring the Clients . . . . .	188
5.5	Configuring the Access Gateway for Kerberos Authentication . . . . .	190
<b>6</b>	<b>Defining Shared Settings</b>	<b>191</b>
6.1	Configuring Attribute Sets . . . . .	191
6.2	Editing Attribute Sets . . . . .	194
6.3	Configuring User Matching Expressions . . . . .	194
6.4	Adding Custom Attributes . . . . .	196
6.4.1	Creating Shared Secret Names . . . . .	196
6.4.2	Creating LDAP Attribute Names . . . . .	197
6.5	Adding Authentication Card Images . . . . .	199
6.6	Creating an Image Set . . . . .	199
6.7	Metadata Repositories . . . . .	200
6.7.1	Creating Metadata Repositories . . . . .	200
6.7.2	Reimporting Metadata Repositories . . . . .	201
<b>7</b>	<b>Configuring SAML and Liberty Trusted Providers</b>	<b>203</b>
7.1	Understanding the Trust Model . . . . .	204
7.1.1	Identity Providers and Consumers . . . . .	204
7.1.2	Embedded Service Providers . . . . .	205
7.1.3	Configuration Overview . . . . .	205
7.2	Configuring General Provider Options . . . . .	206
7.2.1	Configuring the General Identity Provider Options . . . . .	206
7.2.2	Configuring the General Identity Consumer Options . . . . .	207
7.2.3	Configuring the Introductions Class . . . . .	208
7.2.4	Configuring the Trust Levels Class . . . . .	209

7.3	Managing Trusted Providers	209
7.3.1	Creating a Trusted Service Provider for SAML 2.0	210
7.3.2	Creating a Trusted Service Provider for SAML 1.1 and Liberty	212
7.3.3	Creating a Trusted Identity Provider	214
7.3.4	Creating Identity Providers and Service Providers	215
7.4	Modifying a Trusted Provider	216
7.5	Executing Authorization Based Roles Policy During SAML 2.0 Service Provider Initiated Request	217
7.6	Contracts Assigned to SAML 2.0 Service Provider	218
7.7	Configuring Communication Security	220
7.7.1	Configuring Communication Security for Liberty and SAML 1.1	221
7.7.2	Configuring Communication Security for a SAML 2.0 Identity Provider	222
7.7.3	Configuring Communication Security for a SAML 2.0 Service Provider	223
7.8	Selecting Attributes for a Trusted Provider	224
7.8.1	Configuring the Attributes Obtained at Authentication	224
7.8.2	Configuring the Attributes Sent with Authentication	225
7.8.3	Sending Attributes to the Embedded Service Provider	226
7.9	Managing Metadata	227
7.9.1	Viewing and Reimporting a Trusted Provider's Metadata	227
7.9.2	Viewing Trusted Provider Certificates	228
7.9.3	Editing a SAML 1.1 Identity Provider's Metadata	228
7.9.4	Editing a SAML 1.1 Service Provider's Metadata	229
7.9.5	Editing a SAML 2.0 Service Provider's Metadata	231
7.10	Configuring an Authentication Request for an Identity Provider	232
7.10.1	Configuring a Liberty Authentication Request	232
7.10.2	Configuring a SAML 2.0 Authentication Request	234
7.11	Configuring an Authentication Response for a Service Provider	237
7.11.1	Configuring the Liberty Authentication Response	237
7.11.2	Configuring the SAML 2.0 Authentication Response	239
7.11.3	Configuring the SAML 1.1 Authentication Response	241
7.12	Routing to an External Identity Provider Automatically	242
7.13	Defining Options for Liberty or SAML 2.0	242
7.13.1	Defining Options for SAML 2.0 Identity Provider	242
7.13.2	Defining Options for Liberty or SAML 2.0 Service Provider	243
7.13.3	Defining Options for Liberty Identity Provider	244
7.14	Defining Options for SAML 1.1 Service Provider	245
7.15	Defining Session Synchronization for the A-Select SAML 2.0 Identity Provider	245
7.16	Configuring the Liberty or SAML 2.0 Session Timeout	246
7.16.1	Session Termination	247
7.17	Managing the Authentication Card of an Identity Provider	247
7.17.1	Modifying the Authentication Card for Liberty or SAML 2.0	247
7.17.2	Modifying the Authentication Card for SAML 1.1	248
7.18	Using the Intersite Transfer Service	248
7.18.1	Understanding the Intersite Transfer Service URL	249
7.18.2	Specifying the Intersite Transfer Service URL for the Login URL Option	251
7.18.3	Using Intersite Transfer Service Links on Web Pages	252
7.18.4	Configuring an Intersite Transfer Service Target for a Service Provider	254
7.18.5	Configuring Whitelist of Target URLs	255
7.18.6	Federation Entries Management	255
7.18.7	Step up Authentication Example for an Identity Provider Initiated Single Sign-On Request	255
7.19	Enabling or Disabling SAML Tags	257
7.20	Sample Configurations	259
7.20.1	Setting Up Google Applications	259
7.20.2	Setting Up Office 365 Services	261
7.20.3	Integrating Salesforce With Access Manager By Using SAML 2.0	261
7.20.4	Integrating Shibboleth Identity Provider With Access Manager	263

7.21	Configuring Multiple SAML 2.0 Service Providers on the Same Host for a Single SAML Identity Provider	264
<b>8</b>	<b>Configuring WS Federation</b>	<b>265</b>
8.1	Using the Identity Server as an Identity Provider for ADFS	265
8.1.1	Configuring the Identity Server	266
8.1.2	Configuring the ADFS Server	271
8.1.3	Logging In	273
8.1.4	Troubleshooting	274
8.2	Using the ADFS Server as an Identity Provider for an Access Manager Protected Resource	275
8.2.1	Configuring the Identity Server as a Service Provider	276
8.2.2	Configuring the ADFS Server to Be an Identity Provider	279
8.2.3	Logging In	280
8.2.4	Additional WS Federation Configuration Options	280
8.3	Managing WS Federation Providers	281
8.3.1	Creating an Identity Provider for WS Federation	281
8.3.2	Creating a Service Provider for WS Federation	282
8.4	Modifying a WS Federation Identity Provider	282
8.4.1	Renaming the Trusted Provider	282
8.4.2	Configuring the Attributes Obtained at Authentication	283
8.4.3	Modifying the User Identification Method	283
8.4.4	Viewing the WS Identity Provider Metadata	284
8.4.5	Editing the WS Identity Provider Metadata	285
8.4.6	Modifying the Authentication Card	285
8.4.7	Assertion Validity Window	286
8.5	Modifying a WS Federation Service Provider	286
8.5.1	Renaming the Service Provider	286
8.5.2	Configuring the Attributes Sent with Authentication	287
8.5.3	Modifying the Authentication Response	287
8.5.4	Viewing the WS Service Provider Metadata	288
8.5.5	Editing the WS Service Provider Metadata	288
8.6	Configuring STS Attribute Sets	289
8.7	Configuring STS Authentication Methods	289
8.8	Configuring STS Authentication Request	290
<b>9</b>	<b>WS-Trust Security Token Service</b>	<b>291</b>
9.1	Overview	291
9.2	Benefits	293
9.3	Scenarios	293
9.3.1	Scenario 1: Web Service Client Communicating with Token Protected Web Service Provider	293
9.3.2	Scenario 2: Web Single Sign-On and STS	294
9.3.3	Scenario 3: Identity Delegation and Impersonation	295
9.3.4	Renewing a Token	297
9.3.5	Authentication Using SAML Tokens	298
9.4	Configuring WS-Trust STS	300
9.4.1	Enabling WS-Trust	300
9.4.2	Configuring Access Manager for WS-Trust STS	300
9.4.3	Viewing STS Service Details	301
9.5	Configuring Service Providers	302
9.5.1	Adding a Domain and Assigning WS-Trust Operations	302
9.5.2	Adding Web Service Providers	303
9.5.3	Managing Service Provider Domains	305
9.5.4	Managing Service Providers	305
9.5.5	Modifying Service Providers	305
9.5.6	A Sample WS-Policy for Web Service Providers	306

9.6	Configuring Web Service Clients . . . . .	307
9.6.1	Configuring Apache CXF-based Web Service Clients . . . . .	307
9.6.2	Configuring Metro-based Web Service Clients . . . . .	308
9.7	Renew Token - Sample Request and Response . . . . .	309
9.7.1	Renew Token - Sample Request . . . . .	309
9.7.2	Renew Token - Sample Response . . . . .	311
<b>10</b>	<b>Configuring Active Directory Federation Services with SAML 2.0</b>	<b>315</b>
10.1	Prerequisites and Requirements . . . . .	315
10.1.1	Linux Environment . . . . .	316
10.1.2	IP Connectivity . . . . .	316
10.1.3	Name Resolution . . . . .	316
10.1.4	Clock Synchronization . . . . .	316
10.2	Configuring Access Manager as a Claims or Identity Provider and AD FS 2.0 as Relying Party or Service Provider . . . . .	317
10.2.1	Configuring Access Manager . . . . .	317
10.2.2	Configuring AD FS 2.0 . . . . .	319
10.2.3	Example Scenario: Access Manager as the Claims Provider and AD FS 2.0 as the Relying Party . . . . .	324
10.3	Configuring AD FS 2.0 as the Claims or Identity Provider and Access Manager as the Relying Party or Service Provider . . . . .	325
10.3.1	Configuring Access Manager . . . . .	325
10.3.2	Configuring AD FS 2.0 . . . . .	326
10.4	AD FS 2.0 Basics . . . . .	329
10.4.1	Configuring the Token-Decrypting Certificate . . . . .	329
10.4.2	Adding CA Certificates to AD FS 2.0 . . . . .	329
10.4.3	Debugging AD FS 2.0 . . . . .	329
10.5	Troubleshooting . . . . .	329
<b>11</b>	<b>Configuring Single Sign-On for Office 365 Services</b>	<b>331</b>
11.1	Passive and Active Authentication . . . . .	331
11.2	Configuring Active and Passive Authentication By Using WS-Trust and WS-Federation Protocols . . . . .	332
11.2.1	Prerequisite . . . . .	332
11.2.2	Configuring an Office 365 Domain By Using WS-Trust Protocol . . . . .	332
11.2.3	Configuring an Office 365 Domain to Federate with Access Manager . . . . .	333
11.3	Configuring an Office 365 Domain That Supports Passive Federation Using SAML 2.0 Protocol . . . . .	335
11.3.1	Prerequisite . . . . .	335
11.3.2	Configuring an Office 365 Domain to Federate with Access Manager . . . . .	335
11.4	Useful Resources . . . . .	339
11.5	Troubleshooting Scenarios . . . . .	340
11.5.1	WS-Trust and WS-Federation Scenarios . . . . .	340
11.5.2	SAML 2.0 Scenarios . . . . .	341
11.5.3	Office 365 Domain Scenarios . . . . .	341
11.6	Sample Tokens . . . . .	343
11.6.1	Sample SAML Token . . . . .	343
11.6.2	Sample WS-Trust Token . . . . .	345
11.6.3	Sample WS-Federation Token . . . . .	347
<b>12</b>	<b>SP Brokering</b>	<b>349</b>
12.1	Overview . . . . .	349
12.2	Configuring the SP Broker . . . . .	351
12.3	Configuring a Brokering for Authorization of Service Providers . . . . .	354



12.4	Creating and Viewing Brokering Groups	355
12.4.1	Creating a Brokering Group	356
12.4.2	Configuring Trusted Identity Providers and Service Providers	357
12.4.3	Configuring Brokering Rules	359
12.4.4	Constructing Brokering URLs	361
12.4.5	Validating Brokering Rules	363
12.5	Generating the Brokering URLs by Using an ID and Target in the Intersite Transfer Service	365
12.6	Assigning The Local Roles Based On Remote Roles And Attributes	365
12.7	SP Brokering Example	367
<b>13</b>	<b>Configuring User Identification Methods for Federation</b>	<b>369</b>
13.1	Defining User Identification for Liberty and SAML 2.0	369
13.1.1	Selecting a User Identification Method for Liberty or SAML 2.0	369
13.1.2	Configuring the Attribute Matching Method for Liberty or SAML 2.0	372
13.2	Defining User Identification for SAML 1.1	373
13.2.1	Selecting a User Identification Method for SAML 1.1	373
13.2.2	Configuring the Attribute Matching Method for SAML 1.1	374
13.3	Defining the User Provisioning Method	375
13.4	User Provisioning Error Messages	379
<b>14</b>	<b>Configuring Communication Profiles</b>	<b>381</b>
14.1	Configuring a Liberty Profile	381
14.2	Configuring a SAML 1.1 Profile	382
14.3	Configuring a SAML 2.0 Profile	382
<b>15</b>	<b>Configuring Liberty Web Services</b>	<b>385</b>
15.1	Configuring the Web Services Framework	385
15.2	Managing Web Services and Profiles	386
15.2.1	Modifying Service and Profile Details for Employee, Custom, and Personal Profiles	387
15.2.2	Modifying Details for Authentication, Discovery, LDAP, and User Interaction Profiles	389
15.2.3	Editing Web Service Descriptions	390
15.2.4	Editing Web Service Policies	391
15.2.5	Create Web Service Type	393
15.3	Configuring Credential Profile Security and Display Settings	394
15.4	Customizing Attribute Names	396
15.5	Configuring the Web Service Consumer	396
15.6	Mapping LDAP and Liberty Attributes	397
15.6.1	Configuring One-to-One Attribute Maps	398
15.6.2	Configuring Employee Type Attribute Maps	401
15.6.3	Configuring Employee Status Attribute Maps	402
15.6.4	Configuring Postal Address Attribute Maps	404
15.6.5	Configuring Contact Method Attribute Maps	405
15.6.6	Configuring Gender Attribute Maps	407
15.6.7	Configuring Marital Status Attribute Maps	408
<b>16</b>	<b>Maintaining an Identity Server</b>	<b>411</b>
16.1	Managing an Identity Server	411
16.1.1	Updating an Identity Server Configuration	412
16.1.2	Restarting the Identity Server	413
16.2	Editing Server Details	413
16.3	Configuring Component Logging	414
16.3.1	Enabling Component Logging	414
16.3.2	Managing Log File Size	416

16.4	Configuring Session-Based Logging	416
16.4.1	Creating the Administrator Class, Method, and Contract	417
16.4.2	Creating the Logging Session Class, Method, and Contract	418
16.4.3	Enabling Basic Logging	419
16.4.4	Responding to an Incident	420
16.5	Monitoring the Health of an Identity Server	422
16.5.1	Health States	423
16.5.2	Viewing the Health Details of an Identity Server	423
16.5.3	Viewing the Health Details of a Cluster	426
16.6	Monitoring Identity Server Statistics	427
16.6.1	Application	428
16.6.2	Authentications	428
16.6.3	Incoming HTTP Requests	429
16.6.4	Outgoing HTTP Requests	430
16.6.5	Liberty	430
16.6.6	SAML 1.1	431
16.6.7	SAML 2	431
16.6.8	WSF (Web Services Framework)	431
16.6.9	Clustering	433
16.6.10	LDAP	434
16.6.11	SP Brokering	434
16.7	Monitoring API for the Identity Server Statistics	435
16.7.1	Endpoints of the REST API	435
16.7.2	Supported Commands and Their Outputs	436
16.8	Enabling Identity Server Audit Events	449
16.9	Monitoring Identity Server Alerts	452
16.10	Viewing the Command Status of the Identity Server	452
16.10.1	Viewing the Status of Current Commands	453
16.10.2	Viewing Detailed Command Information	453

## **17 Troubleshooting Identity Server and Authentication 455**

17.1	Useful Networking Tools for Linux Identity Server	456
17.2	Troubleshooting 100101043 and 100101044 Liberty Metadata Load Errors	456
17.2.1	The Metadata	457
17.2.2	DNS Name Resolution	458
17.2.3	Certificate Names	459
17.2.4	Certificates in the Required Trust Stores	460
17.2.5	Enabling Debug Logging	461
17.2.6	Testing Whether the Provider Can Access the Metadata	463
17.2.7	Manually Creating Any Auto-Generated Certificates	463
17.3	Authentication Issues	464
17.3.1	Authentication Classes and Duplicate Common Names	464
17.3.2	General Authentication Troubleshooting Tips	464
17.3.3	Slow Authentication	465
17.3.4	Federation Errors	465
17.3.5	Mutual Authentication Troubleshooting Tips	465
17.3.6	Browser Hangs in an Authentication Redirect	466
17.3.7	Duplicate Set-Cookie Headers	466
17.4	Problems Reading Keystores after Identity Server Re-installation	466
17.5	After Setting Up the User Store to Use SecretStore, Users Report 500 Errors	467
17.6	When Multiple Browser Logout Option Is Enabled User Is Not Getting Logged Out From Different Sessions	467
17.7	302 Redirect to 'RelayState' URL after consuming a SAML Response is being sent to an incorrect URL	467
17.8	Configuring SAML 1.1 Identity Provider Without Specifying Port in the Login URL Field	468
17.9	Attributes are Not Available Through Form Fill When OIOSAML Is Enabled	468

17.10	Issue in Importing Metadata While Configuring Identity Provider or Service Provider Using Metadata URL .....	468
17.11	Enabling Secure or HTTPOnly Flags for Cluster Cookies .....	468
17.12	Apache Portable Runtime Native Library Does Not Get Loaded in Tomcat .....	469
17.13	Metadata Mentions Triple Des As Encryption Method .....	471
17.14	Issue in Accessing Protected Resources with External Identity Provider When Both Providers Use Same Cookie Domain .....	471
17.15	SAML Intersite Transfer URL Setup Does Not Work for Non-brokered Setups After Enabling SP Brokering .....	471
17.16	Orphaned Identity Objects .....	472
17.17	Users cannot Log In to Identity Provider When They Access Protected Resources With Any Contract Assigned .....	473
17.18	Attribute Query from OIOSAML.SP Java Service Provider Fails with Null Pointer .....	473
17.19	Disabling the Certificate Revocation List Checking .....	473
17.20	Step up Authentication for the Identity Server Initiated SSO to External Provider Does not Work Unless It has a Matching Local Contract .....	473
17.21	Metadata Cannot be Retrieved from the URL .....	473
17.22	Requesting the Authentication to a Service Provider Fails .....	474
17.23	SAML 2.0 POST Compression Failure Does not Throw a Specific Error Code .....	474
17.24	SAML 1.1 Service Provider Re-requests for Authentication .....	474
17.25	Issue in Generating WS-Federation Claim for SharePoint 2010 On Windows .....	474
17.26	LDAP Authentication Fails while Accessing the Secret Store .....	475
17.27	The Identity Server Statistics Logs Do Not Get Written In Less Than One Minute .....	475
17.28	No Error Message Is Written in the Log File When an Expired Certificate Is Used for the X509 Authentication .....	476
17.29	Terminating an Existing Authenticated User from the Identity Server .....	476
17.30	Clustered Nodes Looping Due to JGroup Issues .....	477
17.31	Authentication With Aliases Fail .....	477
<b>A</b>	<b>About Liberty</b>	<b>479</b>
<b>B</b>	<b>Understanding How Access Manager Uses SAML</b>	<b>481</b>
B.1	Attribute Mapping with Liberty .....	481
B.2	Trusted Provider Reference Metadata .....	482
B.3	Identity Federation .....	482
B.4	Services .....	482
B.5	What's New in SAML 2.0? .....	482
B.6	Identity Provider Process Flow .....	483
B.7	SAML Service Provider Process Flow .....	485
<b>C</b>	<b>Data Model Extension XML</b>	<b>487</b>
C.1	Elements .....	487
C.2	Writing Data Model Extension XML .....	490
<b>D</b>	<b>Configuring the Supported Social Authentication Providers for API Keys and API Secrets</b>	<b>493</b>
D.1	Integrating Access Manager with Facebook .....	493
D.2	Integrating Access Manager with LinkedIn .....	494
D.3	Integrating Access Manager with Twitter .....	494

D.4	Integrating Access Manager with Google+ .....	495
-----	---	-----

---

# About NetIQ Corporation

We are a global, enterprise software company, with a focus on the three persistent challenges in your environment: Change, complexity and risk—and how we can help you control them.

## Our Viewpoint

### **Adapting to change and managing complexity and risk are nothing new**

In fact, of all the challenges you face, these are perhaps the most prominent variables that deny you the control you need to securely measure, monitor, and manage your physical, virtual, and cloud computing environments.

### **Enabling critical business services, better and faster**

We believe that providing as much control as possible to IT organizations is the only way to enable timelier and cost effective delivery of services. Persistent pressures like change and complexity will only continue to increase as organizations continue to change and the technologies needed to manage them become inherently more complex.

## Our Philosophy

### **Selling intelligent solutions, not just software**

In order to provide reliable control, we first make sure we understand the real-world scenarios in which IT organizations like yours operate — day in and day out. That's the only way we can develop practical, intelligent IT solutions that successfully yield proven, measurable results. And that's so much more rewarding than simply selling software.

### **Driving your success is our passion**

We place your success at the heart of how we do business. From product inception to deployment, we understand that you need IT solutions that work well and integrate seamlessly with your existing investments; you need ongoing support and training post-deployment; and you need someone that is truly easy to work with — for a change. Ultimately, when you succeed, we all succeed.

## Our Solutions

- ♦ Identity & Access Governance
- ♦ Access Management
- ♦ Security Management
- ♦ Systems & Application Management
- ♦ Workload Management
- ♦ Service Management

## Contacting Sales Support

For questions about products, pricing, and capabilities, contact your local partner. If you cannot contact your partner, contact our Sales Support team.

<b>Worldwide:</b>	<a href="http://www.netiq.com/about_netiq/officelocations.asp">www.netiq.com/about_netiq/officelocations.asp</a>
<b>United States and Canada:</b>	1-888-323-6768
<b>Email:</b>	<a href="mailto:info@netiq.com">info@netiq.com</a>
<b>Web Site:</b>	<a href="http://www.netiq.com">www.netiq.com</a>

## Contacting Technical Support

For specific product issues, contact our Technical Support team.

<b>Worldwide:</b>	<a href="http://www.netiq.com/support/contactinfo.asp">www.netiq.com/support/contactinfo.asp</a>
<b>North and South America:</b>	1-713-418-5555
<b>Europe, Middle East, and Africa:</b>	+353 (0) 91-782 677
<b>Email:</b>	<a href="mailto:support@netiq.com">support@netiq.com</a>
<b>Web Site:</b>	<a href="http://www.netiq.com/support">www.netiq.com/support</a>

## Contacting Documentation Support

Our goal is to provide documentation that meets your needs. If you have suggestions for improvements, click **Add Comment** at the bottom of any page in the HTML versions of the documentation posted at [www.netiq.com/documentation](http://www.netiq.com/documentation). You can also email [Documentation-Feedback@netiq.com](mailto:Documentation-Feedback@netiq.com). We value your input and look forward to hearing from you.

## Contacting the Online User Community

Qmunity, the NetIQ online community, is a collaborative network connecting you to your peers and NetIQ experts. By providing more immediate information, useful links to helpful resources, and access to NetIQ experts, Qmunity helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, visit <http://community.netiq.com>.

---

# About this Book and the Library

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 [NetIQ Access Manager 4.0 SP1 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.

## Intended Audience

This book 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)

## Other Information in the Library

Before proceeding, you should be familiar with the [NetIQ Access Manager 4.0 SP1 Installation Guide](#) and the [NetIQ Access Manager 4.0 SP1 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”](#).

For conceptual information about Liberty, and to learn about what is new for SAML 2.0, see [Appendix A, “About Liberty,” on page 479](#) and [Appendix B, “Understanding How Access Manager Uses SAML,” on page 481](#).

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

- ♦ [NetIQ Access Manager 4.0 SP1 Administration Console Guide](#)
- ♦ [NetIQ Access Manager 4.0 SP1 Access Gateway Guide](#)
- ♦ [NetIQ Access Manager 4.0 SP1 Policy Guide](#)

- ♦ [NetIQ Access Manager 4.0 SSL VPN Server Guide](#)
- ♦ [NetIQ Access Manager 4.0 SP1Event Codes](#)

---

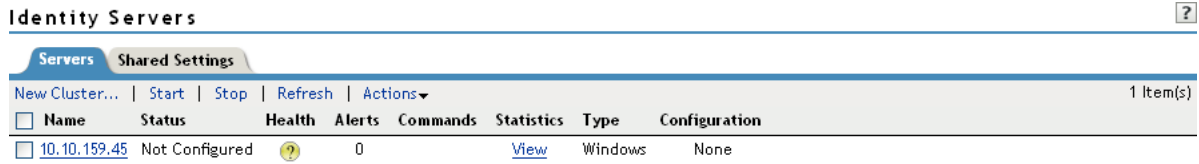
**NOTE:** Contact [namsdk@netiq.com](mailto:namsdk@netiq.com) for any query related to Access Manager SDK.

---



# 1 Configuring an Identity Server

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.



Identity Servers								
Servers		Shared Settings						
New Cluster...		Start	Stop	Refresh	Actions			
<input type="checkbox"/>	Name	Status	Health	Alerts	Commands	Statistics	Type	Configuration
<input type="checkbox"/>	10.10.159.45	Not Configured		0		<a href="#">View</a>	Windows	None

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.

- [Section 1.1, “Managing a Cluster Configuration,” on page 17](#)
- [Section 1.2, “Enabling Role-Based Access Control,” on page 28](#)
- [Section 1.3, “Enabling External Attributes Policy,” on page 28](#)
- [Section 1.4, “Configuring Secure Communication on the Identity Server,” on page 29](#)
- [Section 1.5, “Security Considerations,” on page 34](#)
- [Section 1.6, “Translating the Identity Server Configuration Port,” on page 39](#)
- [Section 1.7, “Using netHSM for the Signing Key Pair,” on page 44](#)

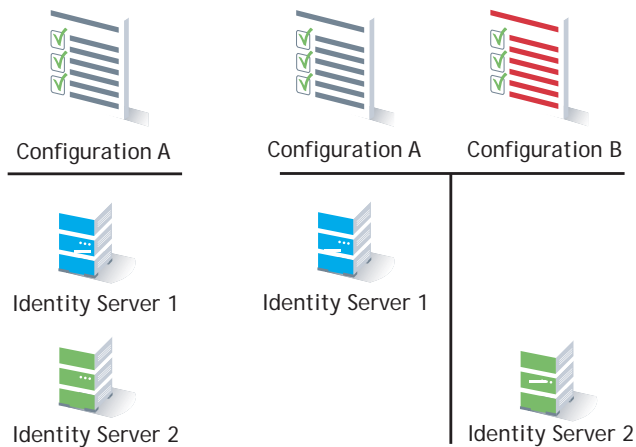
For information about configuring local authentication options, see the following:

- [Chapter 3, “Configuring Local Authentication,” on page 105](#)
- [Chapter 4, “Configuring Advanced Local Authentication Procedures,” on page 145](#)
- [Chapter 6, “Defining Shared Settings,” on page 191](#)
- [Chapter 15, “Configuring Liberty Web Services,” on page 385](#)

## 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



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 18](#)
- ♦ [Section 1.1.2, “Assigning an Identity Server to a Cluster Configuration,” on page 22](#)
- ♦ [Section 1.1.3, “Configuring a Cluster with Multiple Identity Servers,” on page 23](#)
- ♦ [Section 1.1.4, “Configuring Session Failover,” on page 24](#)
- ♦ [Section 1.1.5, “Editing Cluster Details,” on page 25](#)
- ♦ [Section 1.1.6, “Removing a Server from a Cluster Configuration,” on page 26](#)
- ♦ [Section 1.1.7, “Enabling and Disabling Protocols,” on page 27](#)
- ♦ [Section 1.1.8, “Modifying the Base URL,” on page 27](#)

## 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 *NetIQ Access Manager 4.0 SP1 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), by using Liberty, SAML 1.1, SAML 2.0, 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 > Edit**.
- 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 22](#).

- 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 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 1.6, “Translating the Identity Server Configuration Port,” on page 39](#).
  - ♦ If you are configuring a Windows Identity Server, you must also modify the Tomcat `server.xml` file located in 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 1.4.4, “Managing the Keys, Certificates, and Trust Stores,” on page 31](#).

For information about how to replace the `test-connector` certificate, see “[Enabling SSL Communication](#)” in the *NetIQ Access Manager 4.0 SP1 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 you enable this option, it 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.

- ♦ **Deleting Previous User Sessions:** You can configure the Identity Server to delete the previous user sessions if the number of open sessions reaches the maximum limit of allowed sessions that you have specified in the **Limit User Sessions** field. Add `DELETE_OLD_SESSIONS_OF_USER=true` configuration option in the `nidpconfig.properties` configuration file located at `/opt/novell/nids/lib/webapp/WEB-INF/classes` for Linux and `C:\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF\classes` for Windows and restart the Identity Server. This configuration option must be added on all the Identity Servers in the cluster. Previous sessions are cleared across Identity Server clusters only when a fresh authentication request comes in. When the Identity server deletes previous user sessions, it sends a logout request to the service provider through the SOAP back channel.

**Use case:** In this scenario, a user is accessing a protected resource from a machine and wants to access the same protected resource from another device. The Identity Server will not give access to the user if the **Limit User Sessions** has reached a maximum limit. The Identity Server must terminate the old session of the user so that the user can access the new session seamlessly.

**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 or SSL VPN, 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 data between trusted identity providers and service providers and provides the framework for user federation.

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

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

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

**WS-Trust:** Allows secure communication and integration between services by using security tokens.

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

The system displays the Organization page.

Identity Servers ▶

**Create Cluster Configuration** ?

Step 2 of 3: Specify Organization

Name: \*

Display name: \*

URL: \*

**Principal Contact**

Company:

First Name:

Last Name:

Email Address:

Telephone Number:

Contact Type:

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 106](#) for information about this procedure.

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

#### Identity Servers

Servers Shared Settings								
New Cluster...   Start   Stop   Refresh   Actions▼								
<input type="checkbox"/> Name	Status	Health	Alerts	Commands	Statistics	Type	Configuration	
<a href="#">idp-corporate</a>	Current		0		<a href="#">View</a>		<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/> <a href="#">10.10.159.206</a>	Current		0	<a href="#">Complete</a>	<a href="#">View</a>	Linux		

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 23](#) 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 do not 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 *NetIQ Access Manager 4.0 SP1 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 18](#) for information about creating an Identity Server configuration.
  - ◆ See [Section 1.1.2, “Assigning an Identity Server to a Cluster Configuration,” on page 22](#) 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 206](#)), 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 24](#).
  - ☐ To modify the name of the cluster or edit communication details, see [Section 1.1.5, “Editing Cluster Details,” on page 25](#).

## 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” on page 24](#)
- ♦ [“Configuring Session Failover” on page 24](#)
- ♦ [“How Failover Peers Are Selected” on page 25](#)

### 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.

#### Details

To modify the cluster name or its settings, perform the following steps:

- 1 In the Administration Console, click **Devices > Identity Servers > [Name of Cluster] > Details > Edit**.

- 2 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 24](#).

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

## Health

Click **Devices > Identity Servers > [Name of Cluster] > Health**. The Cluster Details page displays health of the cluster and health of each member of the cluster.

## Alerts

Click **Devices > Identity Servers > [Name of Cluster] > Alerts**. The Cluster Details page displays alerts generated by members of the cluster.

To view more details about alerts of each member, click the server under the **Server Name** column.

## Statistics

On the Cluster Statistics page, you can configure the list of statistics to show the desired statistics for an Identity Provider cluster. See [Section 16.6, “Monitoring Identity Server Statistics,” on page 427](#) for the complete list of statistics for each server in an Identity Server cluster.

To configure the statistics, perform the following steps:

- 1 In the Administration Console, click **Devices > Identity Servers > [Name of Cluster] > Statistics > Configure**.
- 2 Select **Set default statistics under "Selected Statistics"** if you want to replace the selected statistics with the default statistics.

The default statistics include Free Memory, Provided Authentications Failures, Consumed Authentications Failures, Cached Sessions, Currently Active Requests - Incoming HTTP Requests, and Currently Active Requests - Outgoing HTTP Requests.

- 3 Select statistics from **Available Statistics** and move to **Selected Statistics**.
- 4 Click **OK**.

The Cluster Statistics page displays the summary of configured statistics for each individual member of the cluster.

To view additional information about a specific Identity Server, click the name of the Identity Server in the **Server Name** column of the summary.

You can also view all the statistics for an individual server of the cluster. Click **View** in the **Statistics** column of the summary to see these additional statistics. For more information, see [Section 16.6, “Monitoring Identity Server Statistics,” on page 427](#).

- 5 Click **Close**.

### 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 16.1, “Managing an Identity Server,” on page 411](#).

---

**IMPORTANT:** When there are only two identity providers in a cluster, then introduce a new identity provider to the cluster and remove the other identity provider.

---

## 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 one of the following commands:

```
/etc/init.d/novell-idp restart  
rcnovell-idp restart
```

- ♦ **Windows Identity Server:** Enter the following commands:

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

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 *NetIQ Access Manager 4.0 SP1 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 policies to the role.

For more information about how to create and configure role policies, see [“Creating Role Policies”](#) in the *NetIQ Access Manager 4.0 SP1 Policy Guide*.

To assign a role 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 Enabling External Attributes Policy

An External Attributes policy must be enabled and configured before users can use the policy for fetching the attributes from external sources.

For more information about how to create and configure External Attributes policies, see [“Creating External Attribute Source Policies”](#) in *NetIQ Access Manager 4.0 SP1 Policy Guide*.

For assigning a policy to users, you must enable it for the Identity Server configuration.

- 1 In Administration Console, click **Devices > Identity Servers > Servers > Edit > External Attributes**.
- 2 Select the check box against the policy name and click **Enable**.
- 3 To disable the policy, select the check box against the policy name and click **Disable**.
- 4 To create a new policy, click **Manage Policies**.
- 5 After enabling or disabling policies, update the Identity Server configuration on the **Servers** tab.

## 1.4 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 *NetIQ Access Manager 4.0 SP1 Setup Guide*.

- ♦ **Signing:** The test-signing (by default) 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 about the services that use the signing certificate, see [“Access Manager Services That Use the Signing Certificate”](#) on page 45.

This certificate can be stored in an external HSM keystore. For information about how to use netHSM to replace and manage this signing certificate, see [Section 1.7, “Using netHSM for the Signing Key Pair,”](#) on page 44.

If you want increased security, you can configure signing and encryption certificate for the service provider. For information, see [Section 1.4.1, “Configuring Enhanced Security for Service Provider Communications,”](#) on page 30

- ♦ **Data Encryption:** The test-encryption (by default) key pair is used to encrypt specific fields or data in the assertions. For more information about the services that use the encryption certificate, see [Section 1.4.3, “Viewing Services That Use the Encryption Key Pair,”](#) on page 31.

If you want increased security, you can configure signing and encryption certificate for the service provider. For information, see [Section 1.4.1, “Configuring Enhanced Security for Service Provider Communications,”](#) on page 30.

To force the browser connections to the Identity Server to support a specific level of encryption, see [Section 1.5.3, “Forcing 128-Bit Encryption,”](#) on page 35.

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 206.

- ♦ **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 207](#).

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 106](#).

This section describes the following tasks:

- ♦ [Section 1.4.1, “Configuring Enhanced Security for Service Provider Communications,” on page 30](#)
- ♦ [Section 1.4.2, “Viewing the Services That Use the Signing Key Pair,” on page 30](#)
- ♦ [Section 1.4.3, “Viewing Services That Use the Encryption Key Pair,” on page 31](#)
- ♦ [Section 1.4.4, “Managing the Keys, Certificates, and Trust Stores,” on page 31](#)

## 1.4.1 Configuring Enhanced Security for Service Provider Communications

When a single identity provider authenticates to multiple service providers, all the assertions are signed by using a common signing key. The assertions are also decrypted by using a common encryption key. Using a single certificate can lead to a vulnerability for all the service providers.

For example, if the common signing or encryption cert is compromised, the information can be used on multiple service providers to potentially gather information.

To mitigate this risk, you can use a single signing and encryption certificate for each service provider.

To define signing and encryption certificate for a service provider, see [Section 1.4.4, “Managing the Keys, Certificates, and Trust Stores,” on page 31](#)

## 1.4.2 Viewing the Services That Use the Signing Key Pair

The following services can be configured to use signing:

- ♦ [“Protocols” on page 30](#)
- ♦ [“SOAP Back Channel” on page 31](#)
- ♦ [“Profiles” on page 31](#)

### 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 Assertions** and **Sign 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 **Trust**.
- 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.

### 1.4.3 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.

### 1.4.4 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 > General > 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. Click **Add** or **Remove** to add or remove the encryption certificates.

**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. Click **Add** or **Remove** to add or remove the signing certificates.

---

**NOTE:** When you change the existing signing/encryption certificate, ensure that the metadata is reimported from an identity or service provider of another cluster.

---

**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:

**Keystore: Provider Introductions SSL Connector**

Keystore name: Provider Introductions SSL Connector

Keystore type: Java

Cluster name: idp-corporate

**Cluster/Configuration Members' Keystores**

Keystore Name	Type	Device
ID Provider Introductions SSL Connector	Java	10.10.159.206

**Certificates**

Replace...

<input type="checkbox"/> Certificate	Alias	Subject
<input type="checkbox"/> <a href="#">jwilson_provo_novell_com</a>	tomcat	CN=jwilson.provo.novell.com

**Replace**

Certificate:

Alias(es):

**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:

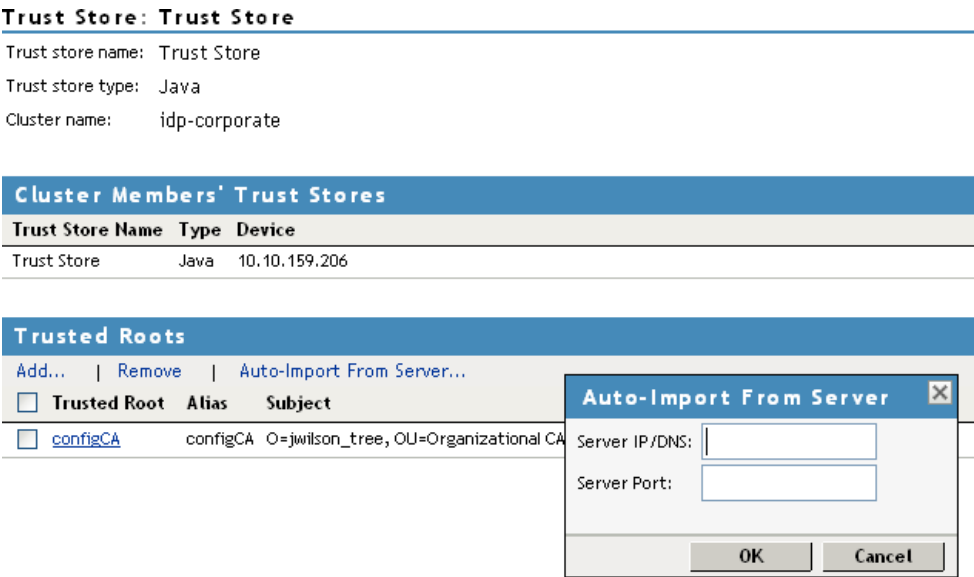
**NIDP Trust Store:** This Identity Server trust store contains the trusted root certificates of all the providers that it trusts. Liberty and SAML 2.0 protocol messages that are exchanged between identity and service providers often need to be digitally signed. A provider uses the signing certificate 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 *NetIQ Access Manager 4.0 SP1 Setup Guide*.

For additional information about managing certificates, see “Security and Certificate Management” in the *NetIQ Access Manager 4.0 SP1 Administration Console Guide*.

## 1.5 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 *NetIQ Access Manager 4.0 SP1 Administration Console Guide*.

Be aware of the following options that can increase security:

- ♦ [Section 1.5.1, “Federation Options,” on page 34](#)
- ♦ [Section 1.5.2, “Authentication Contracts,” on page 34](#)
- ♦ [Section 1.5.3, “Forcing 128-Bit Encryption,” on page 35](#)
- ♦ [Section 1.5.4, “Securing the Identity Server Cookie,” on page 36](#)
- ♦ [Section 1.5.5, “Configuring the Encryption Method for the SAML Assertion,” on page 36](#)
- ♦ [Section 1.5.6, “Configuring SAML 2.0 to Sign Messages,” on page 37](#)
- ♦ [Section 1.5.7, “Blocking Access to Identity Server Pages,” on page 39](#)

### 1.5.1 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 7.10, “Configuring an Authentication Request for an Identity Provider,” on page 232](#).

For more information about how to setup federation, see [“Setting Up Federation”](#) in *NetIQ Access Manager 4.0 SP1 Setup Guide*.

### 1.5.2 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**.

## 1.5.3 Forcing 128-Bit Encryption

All client communication with the Identity Server currently uses 128-bit encryption. If the browser is unable to support 128 bit encryption, the user is not allowed to authenticate. The encryption level supported can be modified by adding or removing the ciphers listed in the `server.xml` file.

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

**Linux:** `/opt/novell/nam/idp/conf`

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

- 2 To edit the `server.xml` entries, search for the `cipher` attribute in the `<Connector>` element and then modify the list of ciphers based on your needs. For example, a sample configuration to enable 128-bit encryption will be as follows:

```
ciphers="SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
TLS_KRB5_WITH_3DES_EDE_CBC_SHA, TLS_KRB5_WITH_RC4_128_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 \(http://java.sun.com/javase/6/docs/technotes/guides/security/SunProviders.html#SunJSSEProvider\)](http://java.sun.com/javase/6/docs/technotes/guides/security/SunProviders.html#SunJSSEProvider).

- 3 To activate the cipher list, restart Tomcat.

**Linux:** Enter one of the following commands:

```
/etc/init.d/novell-idp restart
rcnovell-idp restart
```

**Windows:** Enter the following commands:

```
net stop Tomcat7
```

```
net start Tomcat7
```

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

## 1.5.4 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 *NetIQ Access Manager 4.0 SP1 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:** `/opt/novell/nam/idp/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 one of the following commands:

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

```
rcnovell-idp restart
```

**Windows:** Enter the following commands:

```
net stop Tomcat7
```

```
net start Tomcat7
```

## 1.5.5 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:** `/opt/novell/nam/idp/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-idp restart
rcnovell-idp restart
```

**Windows:** Enter the following commands:

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

The following algorithms for encryption method are supported:

```
<md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
<md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
<md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
<md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep"/>
<md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
```

## 1.5.6 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:** /opt/novell/nam/idp/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 one of the following commands:

```
/etc/init.d/novell-idp restart
rcnovell-idp restart
```

**Windows:** Enter the following commands:

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

You can configure the Identity Server to sign SAML 2.0 post messages for one or multiple trust providers.

To enable the signing of post messages for specific trust providers:

- 1 Open the `nidpconfig.properties` file located in `/opt/novell/nam/idp/webapps/nidp/WEB-INF/classes`.
- 2 Modify the following:

**IS\_SAML2\_POST\_SIGN\_RESPONSE:** This is a global option. Specify true to enable the Identity Provider to send signed saml2 post responses to all its trusted providers.

---

**NOTE:** Configuring `IS_SAML2_POST_SIGN_RESPONSE` is same configuring the `SignPost` in `web.xml`. However, configuring the `nidpconfig.properties` is recommended because it provides more options. These options can be combined with `IS_SAML2_POST_SIGN_RESPONSE` to avoid Access Manager restarts.

---

**SAML2\_POST\_SIGN\_RESPONSE\_TRUSTEDPROVIDERS:** Specify one or more trusted provider's entityID. The Identity Provider will send signed SAML2 post responses to these trusted providers only. When adding more than one provider to the list, separate each entityID with a comma.

## Avoiding Assertion Signing Validation

The `SAML2_AVOID_SIGN_AND_VALIDATE_ASSERTION_TRUSTEDPROVIDERS` flag can be set on the identity provider and the service provider.

If this flag is set on the identity provider, then the complete POST message (excluding the assertion) is signed.

If this flag is set on the service provider, then the signature of the POST message is verified but the assertion signature is not verified. If this flag is not set on the service provider and if the Identity Server sends an assertion where only the assertion is signed, the service provider cannot validate it.

To avoid the assertion signing or validation, perform the following procedure at identity provider/ service provider.

- 1 Open `nidpconfig.properties` file located in the following path:

**Linux:** `/opt/novell/nids/lib/webapp/WEB-INF/classes`

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

- 2 Add the following flag in file:

`SAML2_AVOID_SIGN_AND_VALIDATE_ASSERTION_TRUSTEDPROVIDERS = entityID of the identity provider, entityID2 of the service provider.`

---

**NOTE:** The entityID of the identity provider and the service provider are comma-separated values.

---

For example, to enable this flag on the identity provider use the following format:  
`SAML2_AVOID_SIGN_VALIDATE_ASSERTION_TRUSTEDPROVIDERS = https://www.idp.com:8443/nidp/saml/metadata`

For example, to enable this flag on the service provider use the following format:  
`SAML2_AVOID_SIGN_VALIDATE_ASSERTION_TRUSTEDPROVIDERS = https://www.sp.com:8443/nidp/saml/metadata`

- 3 Save the configuration.
- 4 Restart the Identity Server. For restarting, see [Step 4 on page 37](#).

When this flag is set in the service provider, the SAML 2.0 POST response is signed and assertion is not signed. The service provider will accept the response instead of returning an error.

## 1.5.7 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 140](#)
- [Section 3.6.4, “Blocking Access to the WSDL Services Page,” on page 142](#)

## 1.6 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 1.6.1, “Changing the Port on a Windows Identity Server,” on page 39](#)
- [Section 1.6.2, “Changing the Port on a Linux Identity Server,” on page 40](#)

### 1.6.1 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 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 Tomcat7
net start Tomcat7
```

## 1.6.2 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 40](#).
- ♦ 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 42](#).

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

- ♦ [“Iptable Tutorial 1.2.2”](#)
- ♦ [“NAM Filters for 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 42](#).

- 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](#)” 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](#)” in the *SLES 11 Administration Guide*.

- 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 42](#)
- ♦ [“Adding the SSL VPN Commands” on page 43](#)

### 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 43](#).
  - ♦ 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 42](#).

- 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_deniall` 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_deniall() {
    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
    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

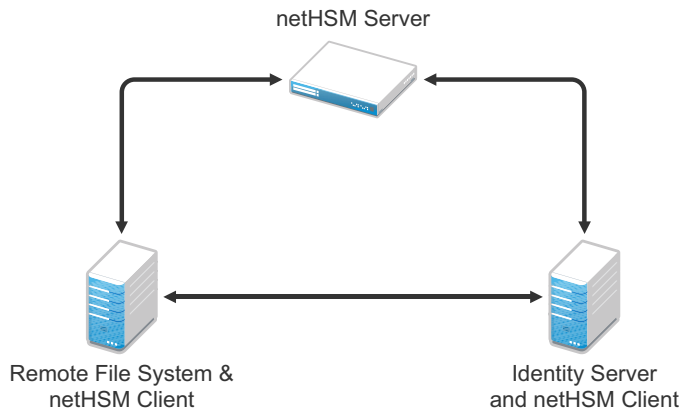
## 1.7 Using netHSM for the Signing Key Pair

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 1.7.1, “Understanding How Access Manager Uses Signing and Interacts with the netHSM Server,” on page 44](#)
- ♦ [Section 1.7.2, “Configuring the Identity Server for netHSM,” on page 46](#)

### 1.7.1 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 45](#)
- ♦ [“Understanding the Interaction of the netHSM Server with Access Manager” on page 46](#)

## Access Manager Services That Use the Signing Certificate

The following services can be configured to use signing:

- ♦ [“Protocols” on page 45](#)
- ♦ [“SOAP Back Channel” on page 45](#)
- ♦ [“Profiles” on page 45](#)

### 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 Assertions** and **Sign 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.

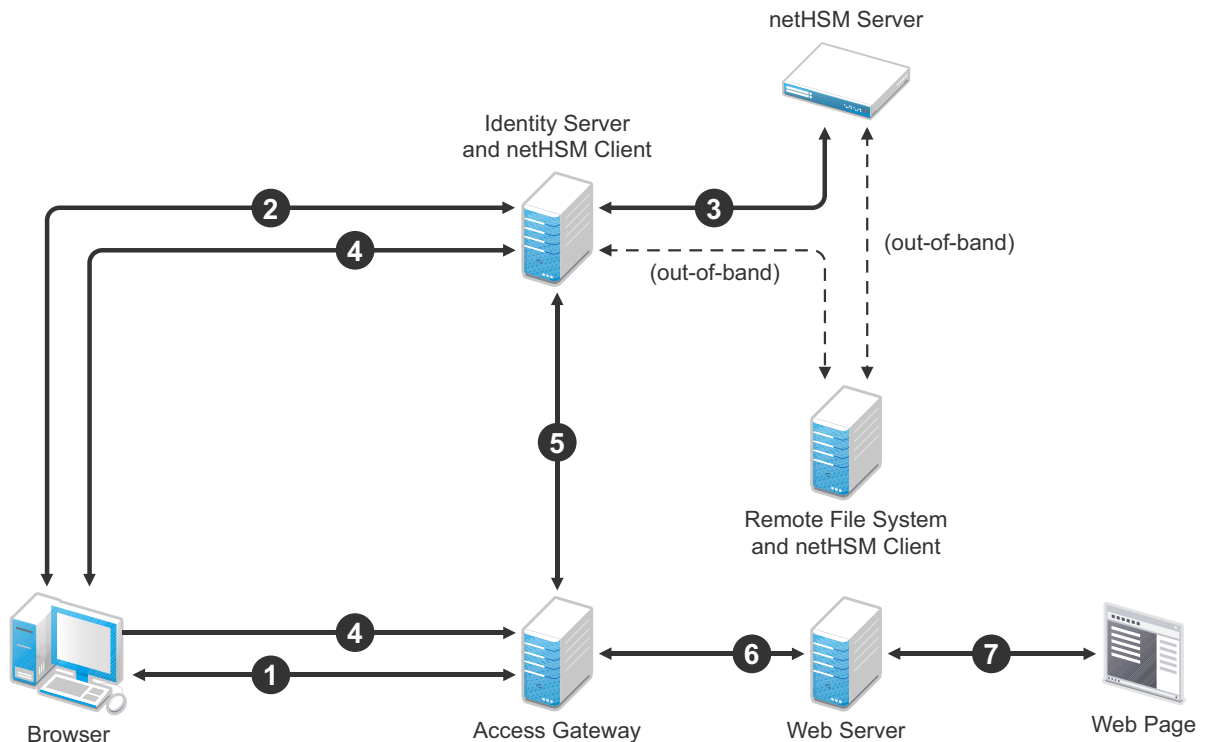
To view your current configuration:

- 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.

### 1.7.2 Configuring the Identity Server for netHSM

- ◆ [“Prerequisites for Using netHSM” on page 47](#)
- ◆ [“Configuring the Identity Server to Be a netHSM Client” on page 47](#)
- ◆ [“Creating the nCipher Signing Key Pair” on page 49](#)
- ◆ [“Configuring the Identity Server to Use the netHSM Certificate” on page 53](#)

- ♦ [“Verifying the Use of the nCipher Key Pair” on page 57](#)
- ♦ [“Troubleshooting the netHSM Configuration” on page 58](#)

## 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 *NetIQ Access Manager 4.0 SP1 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 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 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 one of the following commands:

```
/etc/init.d/novell-idp restart
rcnovell-idp restart
```

**Windows:** Enter the following commands:

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

- 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 49.

## 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.world`.

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.nCipherRSAPrivateEncrypt
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.nCipherRSAPrivateEncrypt
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 <a href="#">Step 1</a> .

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.
-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.

Parameter	Description
-file	The name of the CA certificate file. In this sample configuration, the name is <code>certca.cer</code> .
-keystore	A name for the keystore. In this sample configuration, the name is <code>AMstore.jks</code> .
-storetype	The type of keystore. For nCipher, this must be set to <code>nCipher.sworld</code> .
-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.
<code>sun.security.tools.KeyTool</code>	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 <a href="#">Step 4</a> . In this sample configuration, the name is <code>od93</code> .
-file	The name of the signing certificate file from the CA. In this sample configuration, the name is <code>signcert.der</code> .
-keystore	A name for the keystore. In this sample configuration, the name is <code>AMstore.jks</code> .
-storepass	The password for the keystore. In this sample configuration, the password is <code>mypwd</code> .
-storetype	The type of keystore. For nCipher, this must be set to <code>nCipher.sworld</code> .
-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 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 2008:** `\Program Files (x86)\Novell\devman\jcc\certs\idp`

- 13b Ensure that 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 53.

## 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 54
- ♦ [“Configuring a Windows Identity Server for the Certificate”](#) on page 55

## 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/nam/idp/bin/dtomcat7` 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 `tomcat7.conf` file:

**4a** In a text editor, open the `opt/novell/nam/idp/conf/tomcat7.conf` file.

**4b** Add the following lines:

```
JAVA_OPTS="{JAVA_OPTS} -Dcom.novell.nidp.extern.config.file=
/opt/novell/nids/lib/webapp/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 `/opt/novell/nam/idp/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 <code>com.ncipher.provider.km.nCipherKM</code> .
<provider_name>	The name of the provider. For nCipher, this must be set to <code>nCipherKM</code> .
<keystore_type>	The type of keystore. For nCipher, this must be set to <code>nCipher.sworld</code> .
<keystore_name>	The name you specified when you created the keystore. In this sample configuration, the name is <code>AMstore.jks</code> .
<keystore_pwd>	When you use module-protected keys, the keystore password must be null. For example:  <code>com.novell.nidp.extern.signing.keystorePwd=</code>
<key_alias>	The alias you created for the key when you created the key. In this sample configuration, the name is <code>od93</code> .
<key_pwd>	When you use module-protected keys, the key password must be null. For example:  <code>com.novell.nidp.extern.signing.keyPwd=</code>

- 6 To restart Tomcat, enter one of the following commands:

```
/etc/init.d/novell-idp restart
rcnovell-idp restart
```

- 7 Continue with [“Verifying the Use of the nCipher Key Pair” on page 57](#).

## 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 `tomcat7w.exe` utility located in the following directory:

**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:\nfast\java\classes\rsaprivenc.jar;C:\nfast\java\classes\spp.jar"
```

**2d** Save your changes.

**3** Add the netHSM certificate configuration lines to the `tomcat7.conf` file:

**3a** Run the `tomcat7w.exe` utility located in the following directory:

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

- 5 To restart Tomcat, enter the following commands:

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

- 6 Continue with [“Verifying the Use of the nCipher Key Pair” on page 57](#).

## 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 58](#).

## 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 about how to change the port, see [Step 6 on page 47](#). 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 one of the following commands:

```
/etc/init.d/novell-idp stop  
rcnovell-idp 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 one of the following commands:

```
/etc/init.d/novell-idp start  
rcnovell-idp 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/nam/logs/idp/tomcat` directory.

**4e** Restart Tomcat by entering one the following commands:

```
/etc/init.d/novell-idp restart  
rcnovell-idp restart
```

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

```
tail -f /var/opt/novell/nam/logs/idp/tomcat/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.
- ♦ Ensure that 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 54](#).
- ♦ Verify that the `tomcat7.conf` file is configured correctly. See [Step 4 on page 54](#).

## 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 `tomcat7.conf` file in the `/opt/novell/nam/idp/conf` directory, add the following line:

```
JAVA_OPTS="{JAVA_OPTS} -DJCECSP_DEBUG=255 -DJCECSP_DEBUGFILE=/opt/novell/
nam/idp/logs/nCipher_jcecs debug"
```

**5b** Restart Tomcat by entering one of the following command:

```
/etc/init.d/novell-idp restart
rcnovell-idp restart
```

**5c** Look for clues in the `nCipher_jcecs debug` file.

---

# 2 Customizing Login Pages, Logout Pages, and Messages

This section includes the following topics:

- [Section 2.1, “Customizing the Identity Server Login Page,” on page 61](#)
- [Section 2.2, “Customizing the Identity Server Logout,” on page 80](#)
- [Section 2.3, “Customizing Identity Server Messages,” on page 82](#)
- [Section 2.4, “Sample Custom Login Pages,” on page 86](#)
- [Section 2.5, “Preventing Cross-site Scripting Attacks,” on page 101](#)

## 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 NetIQ 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 62](#).
- 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 73](#).
- 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 79](#).
- You need to sanitize the JSP file to prevent XSS attacks. See, [Section 2.5, “Preventing Cross-site Scripting Attacks,” on page 101](#).

**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.

**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 140](#).

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

- [“Customizing Identity Server Messages” on page 82](#)
- [“Customizing Error Messages and Error Pages on Access Gateway” or “Customizing Error Messages and Error Pages on Access Gateway” in the \*NetIQ Access Manager 4.0 SP1 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 63](#).
- ♦ If you want to maintain the features of the 4.0 page and use its authentication cards but you want to remove the NetIQ 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 65](#).
- ♦ 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 login.jsp File” on page 70](#).
- ♦ 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 70](#)

---

**NOTE:** 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.

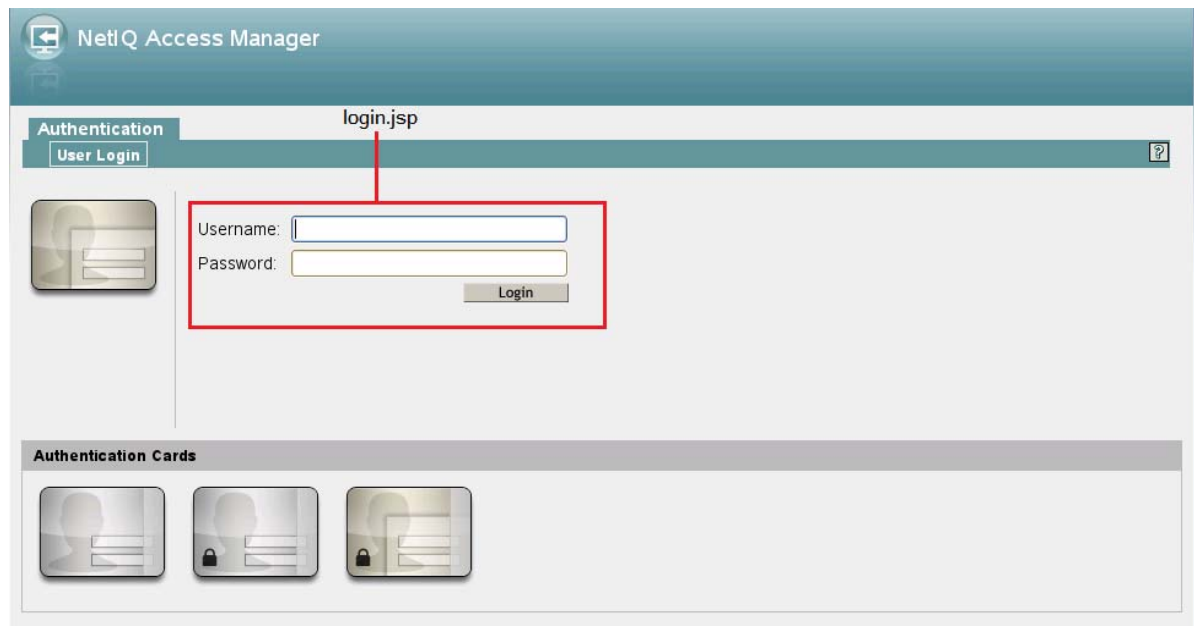
During an upgrade, you can select to restore custom login pages, but NetIQ still recommends that you have your own backup of any customized files.

---

## 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 about how to use this property, see [“Query Property” on page 124](#).

**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 about configuring the other options for a contract, see [Section 3.4, "Configuring Authentication Contracts," on page 128](#).

**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:** `/opt/novell/nids/lib/webapp/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 about how to create a custom message properties file, see [Section 2.3.1, "Customizing Messages," on page 82](#).)

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 using one of the following commands:

**Linux Identity Server:** Enter the following command:

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

```
rcnovell-idp restart
```

**Windows Identity Server:** Enter the following commands:

```
net stop Tomcat7
```

```
net start Tomcat7
```

**8** To view a sample custom page with these modifications, see [Section 2.4.1, “Modified login.jsp File for Credential Prompts,” on page 86](#).

## 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 Access Manager product name and the NetIQ 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 65](#)
- ♦ [“Customizing the Card Display” on page 67](#)
- ♦ [“Customizing the Credential Frame” on page 67](#)
- ♦ [“Customizing the nidp.jsp File to Customize Error Message” on page 69](#)

## 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:** `/opt/novell/nids/lib/webapp/jsp`

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

**2** Replace the header title that appears in the top frame (“NetIQ 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";
```

Ensure that 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 2-1](#)):

**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 NetIQ 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 67](#).
  - ♦ To control the cards displayed in the Authentication Cards section, see [“Customizing the Card Display” on page 67](#).
  - ♦ To configure the Identity Server to use your custom pages, see [“Adding Logic to the main.jsp File” on page 75](#).
  - ♦ To view a sample custom page with these modifications, see [Section 2.4.2, “Custom nidp.jsp File with Custom Credentials,” on page 89](#).

## 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 about modifying the card options, see [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).

Continue with one of the following:

- ♦ To modify what appears in the credential frame, continue with [“Customizing the Credential Frame” on page 67](#)
- ♦ To configure the Identity Server to use your custom pages, see [“Adding Logic to the main.jsp File” on page 75](#).

## 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 about how to use this property, see [“Query Property” on page 124](#).

- 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 about configuring the other options for a contract, see [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).

- 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:** `/opt/novell/nids/lib/webapp/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 about how to create a custom message properties file, see [Section 2.3.1, “Customizing Messages,” on page 82](#).)

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 each Identity Server using one of the following commands:

**Linux Identity Server:** Enter one of the following command:

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

```
rcnovell-idp restart
```

**Windows Identity Server:** Enter the following commands:

```
net stop Tomcat7
```

```
net start Tomcat7
```

- 7 To view a sample custom page with these modifications, see [Section 2.4.2, “Custom nidp.jsp File with Custom Credentials,”](#) on page 89.
- 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 75.

## Customizing the nidp.jsp File to Customize Error Message

The Identity Server publishes a generic error message for error code during SAML failure such as request denied or Invalid Name ID Policy and so on. You can customize the NIDP jsp file available at `/opt/novell/nids/lib/webapp/jsp` and write an appropriate error message for either redirection or to inform the user about the issue with an appropriate message. Perform the following steps to customize error message.

---

**NOTE:** In the following example the specified code snippet is for simulating `InvalidNameIDPolicy` error for SAML 2.0.

---

- 1 Generate an error condition with for example, Invalid Name ID Policy.
- 2 Customized the `nidp.jsp` file and add the following code for redirection.

```

com.novell.nidp.ui.MenuHandler redirectMenuHandler;
    com.novell.nidp.NIDPMessage redirectMessage;
    String redirectCause;

    redirectMenuHandler = new MenuHandler(request, response);
    redirectMessage = redirectMenuHandler.getMessage(true);
    if (redirectMessage != null && redirectMessage instanceof
com.novell.nidp.NIDPError) {
        redirectCause = ((com.novell.nidp.NIDPError)
redirectMessage).getNIDPExceptionMsg();
        System.out.println("***** redirectCause" + redirectCause);
        if (redirectCause != null &&
redirectCause.indexOf("InvalidNameIDPolicy") != -1) {
            response.sendRedirect("http://www.novell.com");
            return;
        }
    }
}

```

- 3 Restart the Identity Server by using the `rcnovell-idp restart` command.
- 4 Verify that when failure occurs, SAML shows the following message in the authentication response.

```

<samlp:Status><samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Responder"><samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy"/></
samlp:StatusCode>

```

Due to customized `nidp.jsp` file, SAML redirects to specified location.

- 5 Rerun the failure and verify that instead of displaying 300101008, nidp page redirects to the specified `www.novell.com` location.

## Modifying the login.jsp File

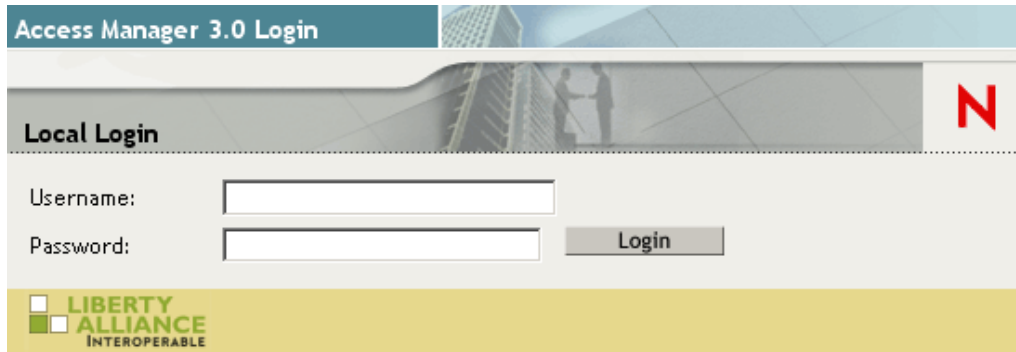
The `login.jsp` file gives you just the credential frame with the login prompts in an `iframe`. It has no branding header. If you use this page, you are responsible for writing the HTML code for the header and the branding.

- 1 Copy the `login.jsp` file and rename it. The JSP files are located on the Identity Server in the following directory:  
**Linux:** `/opt/novell/nids/lib/webapp/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 67](#).
- 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 74](#).

## 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 NetIQ branding and modify the credential prompts.

- ♦ [“Modifying the Branding in the 3.0 Login Page” on page 71](#)
- ♦ [“Modifying the Credentials in the 3.0 Login Page” on page 72](#)

## 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 4.0 Login Page”](#) in the *NetIQ Access Manager 4.0 SP1 Setup 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 4.0 Login Page”](#) in the *NetIQ Access Manager 4.0 SP1 Setup 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"></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 72](#).
- ♦ 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 74](#).

## 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 about how to use this property, see [“Query Property” on page 124](#).

- 1f** Click **OK**.

- 1g** Create a contract that uses this method.

For information about configuring a contract, see [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).

- 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 about how to create a custom message properties file, see [Section 2.3.1, “Customizing Messages,” on page 82.](#))

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 one of the following commands:

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

```
rcnovell-idp restart
```

**Windows Identity Server:** Enter the following commands:

```
net stop Tomcat7
```

```
net start Tomcat7
```

**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 74.](#)

## 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 74.](#)
- ♦ 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 75.](#)

## 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 123](#).
  - 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 ensure that 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 79](#).

## 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.jsp`s that are used by the contract.

Contract	Configuration Details	
Contract1	URI	login1/custom1
	Method1	Configured with the following JSP property:  <b>Property Name:</b> JSP  <b>Property Value:</b> 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

Contract	Configuration Details	
Contract3	Method2	Configured with the following two properties:  <b>Property Name:</b> JSP <b>Property Value:</b> login2  <b>Property Name:</b> Query <b>Property Value:</b> (&(objectclass=person)(mail=%Ecom_User_ID%))
	URI	login3/custom3
	Method3	Configured with the following two properties:  <b>Property Name:</b> JSP <b>Property Value:</b> login3  <b>Property Name:</b> Query <b>Property Value:</b> (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.jsp` 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:** `/opt/novell/nids/lib/webapp/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 79](#).

## Customizing Certificate Errors

If you want the Identity Server to display error details when certificate validation fails, edit the `/opt/novell/nam/idp/conf/server.xml` by using the following procedure:

- 1 Search for the `clientauth` attribute in the `server.xml` file.
- 2 Modify the value of the `clientauth` attribute from the default value of `false` to `want`.
- 3 Save the file and restart Identity Server by using the `rcnovell-idp restart` command.
- 4 Export the certificate from the Administration Console by using the **Security > Certificates** option.
- 5 Import the certificate into the browser:
  - Internet Explorer:** Tools > Internet Option > Content > Certificate > Add into Trusted Roots
  - Mozilla Firefox:** Firefox Options > Encryption > Advanced > View Certificates > Servers.

## 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. Ensure 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:** `/opt/novell/nam/idp/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 80](#)
- [Section 2.2.2, “Replacing the Logout Page with a Custom Page,” on page 80](#)
- [Section 2.2.3, “Configuring for Local Rather Than Global Logout,” on page 81](#)

To customize the logout page when the user logs out of an Access Gateway protected resource, see “[Customizing Logout Requests](#)” in the *NetIQ Access Manager 4.0 SP1 Access Gateway Guide*. When you have both Liberty and SAML 2.0 sessions running on the Identity Server and you log out of the Access Gateway, the `logoutsuccess.jsp` page is not executed with the customization you have made to the logout page. For information about the workaround, see “[Logging Out of Sessions to the Access Gateway and SAML Connectors when Branding Exists in the Customized Logout Page](#)” in the *NetIQ Access Manager 4.0 SP1 Access Gateway Guide*.

---

**NOTE:** After customizing a JSP file, you need to sanitize the JSP file to prevent XSS attacks. See, [Section 2.5, “Preventing Cross-site Scripting Attacks,” on page 101](#).

---

### 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 about how to modify `nidp.jsp` for logos, titles, and colors, see “[Rebranding the Header](#)” on page 65.

---

**IMPORTANT:** Save a copy of your modified `nidp.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:** `/opt/novell/nids/lib/webapp/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:** `/opt/novell/nids/lib/webapp/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

- ♦ [Section 2.3.1, “Customizing Messages,” on page 82](#)
- ♦ [Section 2.3.2, “Customizing the Branding of the Error Page,” on page 84](#)
- ♦ [Section 2.3.3, “Customizing Tooltip Text for Authentication Contracts,” on page 85](#)

---

**NOTE:** After customizing a JSP file, you need to sanitize the JSP file to prevent XSS attacks. See, [Section 2.5, “Preventing Cross-site Scripting Attacks,” on page 101.](#)

---

### 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:** `/opt/novell/nids/lib/webapp/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:** `/opt/novell/nam/idp/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 one of the following commands:

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

```
rcnovell-idp restart
```

- ♦ **Windows Identity Server:** Enter the following commands:

```
net stop Tomcat7
```

```
net start Tomcat7
```

**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 Messages and Error Pages on Access Gateway](#)” in the *NetIQ Access Manager 4.0 SP1 Access Gateway Guide*.

## 2.3.2 Customizing the Branding of the Error Page

The error page (`err.jsp`) is returned when the Identity Server encounters an error with the following message:

```
Error: Unable to authenticate, (300101014-esp-01E79F6000B87D4E8)
```

The file is located in the following directory.

**Linux:** `/opt/novell/nids/lib/webapp/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 about customizing the error message, see [Section 2.3.1, “Customizing Messages,” on page 82](#).

You can customize the following items:

- ♦ The window title and the display title. See “[Customizing the Titles](#)” on page 84.
- ♦ The header image and the Novell logo. See “[Customizing the Images](#)” on page 85.
- ♦ Background colors. See “[Customizing the Colors](#)” on page 85.

### 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></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"></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 the 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 82](#)), 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 one of the following commands:
 

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

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

## 2.4 Sample Custom Login Pages

- ♦ [Section 2.4.1, "Modified login.jsp File for Credential Prompts," on page 86](#)
- ♦ [Section 2.4.2, "Custom nidp.jsp File with Custom Credentials," on page 89](#)
- ♦ [Section 2.4.3, "Custom 3.1 login.jsp File," on page 95](#)
- ♦ [Section 2.4.4, "Custom 3.0 login.jsp File," on page 98](#)

---

**NOTE:** After customizing a JSP file, you need to sanitize the JSP file to prevent XSS attacks. See, [Section 2.5, "Preventing Cross-site Scripting Attacks," on page 101](#).

---

### 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.

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 63](#).

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.getBGCColor()%>" 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%" cellpadding="0"
cellspacing="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>

```

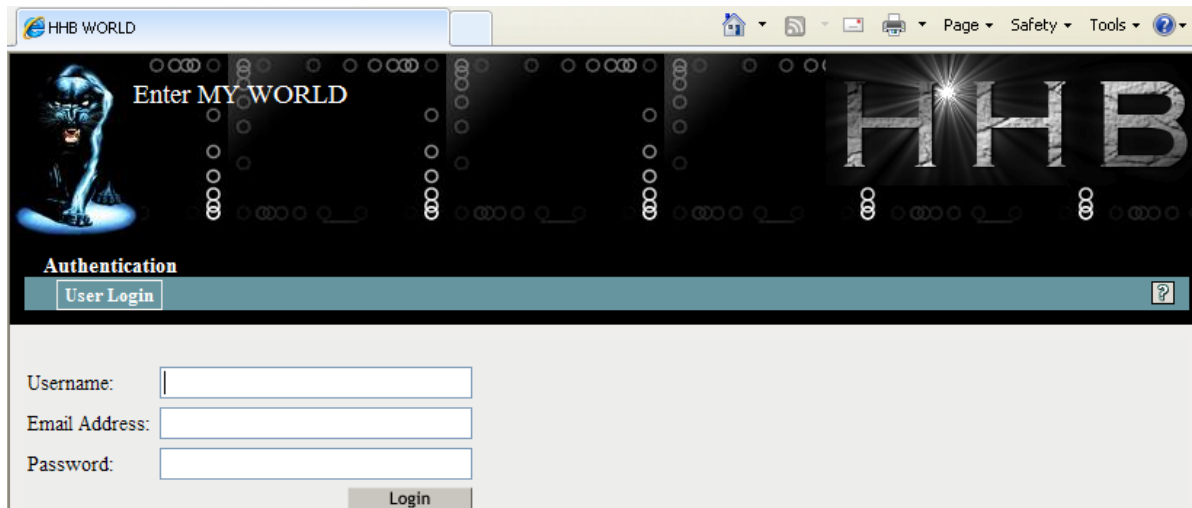


## 2.4.2 Custom nidp.jsp File with Custom Credentials

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 65](#) and [“Adding Logic to the main.jsp File” on page 75](#).

[Figure 2-4](#) illustrates the login page that the following custom `nidp.jsp` file and `main.jsp` file create.

**Figure 2-4** 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 86](#). The following sections provide the other required sample files to create this login page and information about the required method and contract:

- ♦ [“The Modified nidp.jsp File” on page 89](#)
- ♦ [“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:

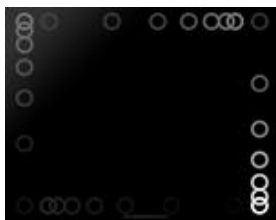
```
// Background color
String bgcolor = "#000000";

// Menu color
String menucolor = "#000000";

// Border color
String bcolor = "#000000";
```

[Figure 2-5](#) illustrates the image (`images2.jpeg`) that this custom page uses for the header background image:

**Figure 2-5** 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 2-6](#) illustrates the image (`images3.jpeg`) that this custom page uses for the product logo that appears on left of the header frame.

**Figure 2-6** Header Image



[Figure 2-7](#) illustrates the image (`hhbimages.jpeg`) that this custom page uses to replace the Novell company logo on the right of the header frame.

**Figure 2-7** 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"
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 .tabls { background-color: <%=menucolor%>; padding-left: 3px;
padding-right: 8px; text-align: center; white-space: nowrap; }
            .NLtab .tabls a { text-decoration: none; }
            .NLtab span.tabls { padding:5; color: white; font-size: 0.9em; font-weight:
bold; line-height: 17px; background-color: transparent; background-image: none;
text-decoration: none; }
            .NLtab .tablu { 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.tablu { 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=yes,wid
th=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 cellpadding=0 width=100% border=0>
            <tr>
            <td width=100%>
                <div id="header"></
div>
                <div id="logo"></
div>
                <div id="title"><%=hdrTitle%></div>
            </td>
            </tr>
            </table>
            </td>
        </tr>
        <tr>
        <td>
            <table cellpadding=5 width=100%>
            <tr>
            <td>
                <%= include file="menus.jsp" %>
            </td>
            </tr>
            </table>
            </td>
        </tr>
        <tr>
        <td>
            <table cellpadding=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>

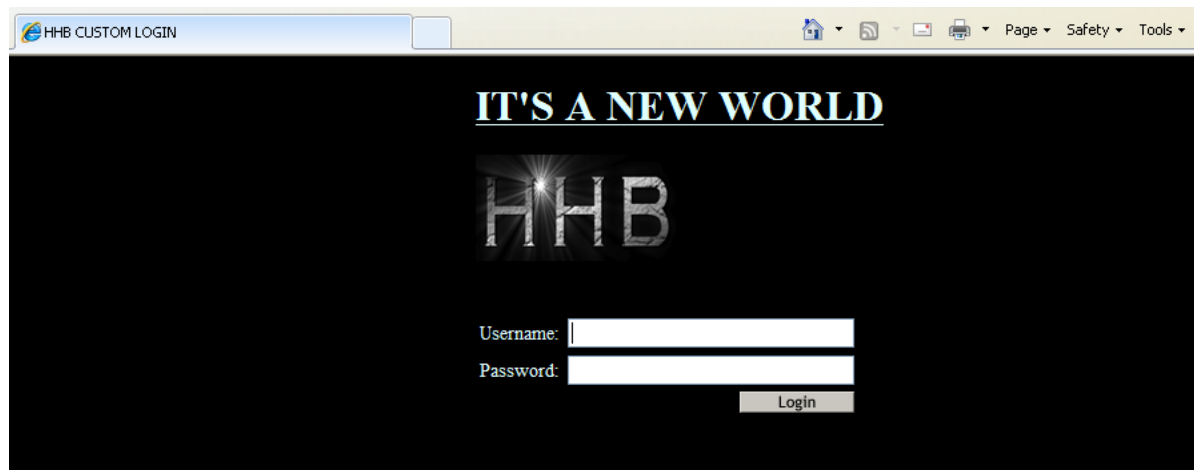
Replace <filename> with the name of your login page that modifies the credential prompts. Do not include the JSP extension in the value.

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 2-8](#) illustrates such a page.

*Figure 2-8 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 97](#)

### The Modified login.jsp File

This custom page does not modify the credential frame. The lines that define the window title (HNB 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" cellpadding="0"
cellpadding="0">
                <tr>
                    <div id="headimage"></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 4.0 Login Page](#)” in the *NetIQ Access Manager 4.0 SP1 Setup Guide*.

Figure 2-9 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 2-9** Custom Page Derived from the 3.0 login.jsp File

The screenshot shows a web browser window with the title 'HHB World'. The browser's address bar and menu bar are visible. The main content area of the page has a header with 'HHB Partner' on the left and a background image of a modern building on the right. Below the header, there is a section titled 'My Company'. Under this section, there are three input fields: 'Username:', 'Email Address:', and 'Password:'. To the right of the 'Password:' field is a 'Login' button. At the bottom of the page, there is a yellow banner with the 'LIBERTY ALLIANCE INTEROPERABLE' logo.

To create this page, see the following sections:

- ♦ “[Modifying the File](#)” on page 99
- ♦ “[The Method and the Contract](#)” on page 101

## 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" cellpadding="0"
cellpadding="0">
                <tr>
                    <td width="50%" height="80 px">&nbsp;</td>
                    <td colspan="2">
                        <div id="title"><b>HHB Partner</b></div>
                        <div id="locallabel"><b>My Company</b></div>
                        <div id="headimage"></div>
                        </td>
                    <td width="100%">&nbsp;</td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

```
</tr>
<tr>
    <td width="50%">&nbsp;</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></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></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">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
                                        <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%">&nbsp;</td>
            </tr>

            <tr>
                <td width="50%"></td>
                <td style="background-color: #E6D88C; padding-left: 10px"></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.

## 2.5 Preventing Cross-site Scripting Attacks

By default, Access Manager does extensive checks to prevent Cross-site Scripting (XSS) attacks. However, Access Manager does not validate a JSP file if you have customized it. If you modify JSP files to customize the login, logout, error pages, and so forth, you must sanitize the JSP file to prevent XSS attacks.

You need to perform either one of the following options to sanitize the customized JSP file:

- ♦ [“Option 1: HTML Escaping” on page 102](#)
- ♦ [“Option 2: Filtering” on page 103](#)

## 2.5.1 Option 1: HTML Escaping

Perform the following XSS checks for the customized JSP file to protect it from possible XSS attacks. For more information about XSS prevention techniques, see [XSS \(Cross Site Scripting\) Prevention Cheat Sheet](#).

Perform the following steps:

- 1 Verify if the `org.apache.commons.lang.StringEscapeUtils` class is available in the JSP file.  
For example, the following import statement should be available in the import section of the JSP file:

```
<%@ page import="org.apache.commons.lang.StringEscapeUtils"%>
```

- 2 Verify if all URL query parameter values are sanitized.

The following code snippet sample shows how URL query parameter values (uname and target) can be sanitized:

```
<%//Fetch the values from URL query parameters
String target = (String) request.getAttribute("target");
String uname = (String) request.getAttribute("username");
String sanitizedUName = "";
if (uname != null){
//Sanitize the value assigned to uname
sanitizedUName = StringEscapeUtils.escapeHtml(uname);
}
String sanitizedTarget = "";
if (target != null){
//Sanitize the value assigned to target query param
sanitizedTarget = StringEscapeUtils.escapeHtml(target);
}%>
```

- 3 Add double quotes (") in value attribute (or any attribute that is dynamically assigned) for any HTML element that get assigned with above URL query param value.

```
<!-- The last 2 double quotes are mandatory to prevent XSS attacks -->
<input type="text" class="smalltext" name="Ecom_User_ID" size="30"
value="<%=sanitizedUName%>">
...
...
<!-- The last 2 double quotes are mandatory to prevent XSS attacks -->
<input type="hidden" name="target" value="<%=sanitizedTarget%>">
```

- 4 Restart the component whose JSP file you have modified. For example, if you modify the Identity Server's JSP file, restart the Identity Server by running the following command:

**Linux:** `sh /etc/init.d/novell-idp restart`

**Windows:** `net start Tomcat7`

## 2.5.2 Option 2: Filtering

This approach might have a minor performance impact due to the checks it performs. If you perform HTML escaping in customized JSP pages, you do not need to perform this additional filtering.

Perform the followings steps to sanitize the Identity Server's customized JSP file:

- 1 Download the `eMFrame_xss.jar` file from ([http://www.netiq.com/documentation/netiqaccessmanager4/resources/eMFrame\\_xss.jar](http://www.netiq.com/documentation/netiqaccessmanager4/resources/eMFrame_xss.jar)).

This library prevents XSS based attacks.

- 2 Place this library at the following location:

**Linux:** `/opt/novell/nids/lib/webapp/WEB-INF/lib`

**Windows:** `\Program Files (x86)\Novell\Tomcat\webapps\nidp\WEB-INF\lib`

- 3 Add a filter in the `web.xml` file located at the following location:

**Linux:** `/opt/novell/nam/idp/webapps/nidp/WEB-INF.`

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

```
<filter>
<filter-name>XSS</filter-name>
<display-name>XSS</display-name>
<description>Filters XSS injections.</description>
<filter-class>com.novell.emframe.fw.filter.CrossScriptingFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>XSS</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

- 4 Restart the Identity Server by running the following command:

**Linux:** `sh /etc/init.d/novell-idp restart`

**Windows:** `net start Tomcat7`



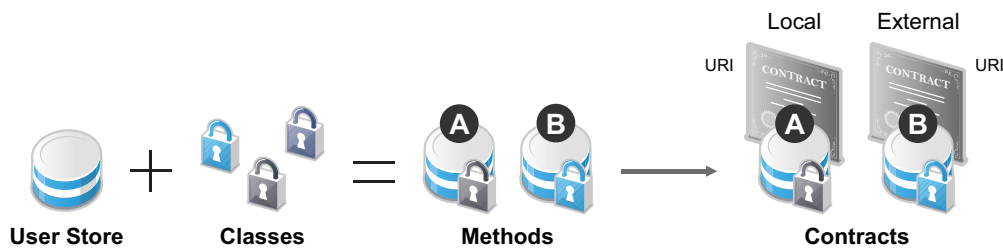


# 3 Configuring Local Authentication

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 106.
- ♦ **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 120.
- ♦ **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 125.
- ♦ **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 128.

This section also explains the following:

- ♦ [“Using a Password Expiration Service”](#) on page 132
- ♦ [“Admin can configure some more parameters that wiUsing Activity Realms”](#) on page 135
- ♦ [“Specifying Authentication Defaults”](#) on page 137
- ♦ [“Managing Direct Access to the Identity Server”](#) on page 139

## 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 107](#).

**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 107](#).

3 Click **OK**, then update the Identity Server if you have modified the configuration.

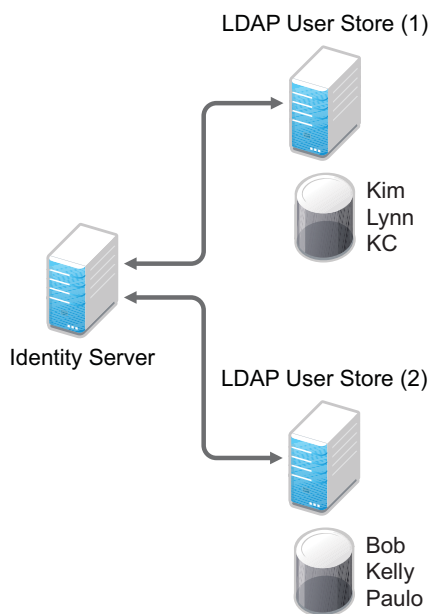
See the following sections for specific configuration tasks:

- [Section 3.1.1, “Using More Than One LDAP User Store,” on page 106](#)
- [Section 3.1.2, “Configuring the User Store,” on page 107](#)
- [Section 3.1.3, “Configuring an Admin User for the User Store,” on page 111](#)
- [Section 3.1.4, “Configuring a User Store for Secrets,” on page 111](#)

### 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 ensure that 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.

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.

## Create Cluster Configuration

### Step 3 of 3: Specify initial User Store

Name: \*

internal

Admin name: \*

cn=admin,o=novell

(Ex: cn=admin,o=novell)

Admin password: \*

XXXXXXXXXX

Confirm password: \*

XXXXXXXXXX

Directory type:

eDirectory

☐ Install NMAAS SAML method
 ☐ Enable Secret Store lock checking

#### LDAP timeout settings

LDAP Operation:

15

seconds

Idle Connection:

10

seconds

#### Server replicas

[New](#) | [Delete](#) | [Validate](#)

<input type="checkbox"/>	Name	IP Address	Port	Use SSL	Max. Connections	Validation Status
No items						

#### Search Contexts

[New](#) | [Delete](#) | [Up](#) | [Down](#)

<input type="checkbox"/>	Context	Scope
No items		

<< Back

Finish

Cancel

### 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 111](#).

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 \(http://developer.novell.com/documentation/nacm31/nacm\\_enu/data/bfg38fg.html\)](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, ensure that 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 111](#).

**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 15.3, “Configuring Credential Profile Security and Display Settings,” on page 394](#).) 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 50. 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 about how to create such contracts, see [Section 17.3.1, "Authentication Classes and Duplicate Common Names," on page 464](#).

---

**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 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, ).
- ♦ 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. Secrets are a way to capture user input like Login ID and password credentials. These input data can later be reused or injected using Form Fill and Identity Injection policies. This feature is especially helpful when your Access Manager Credential Profile does not contain credentials for an application protected by Access Manager yet a single sign-on experience is required. Where and how the secrets can be stored is configurable and depends upon your user store:

- ♦ [“Configuring the Configuration Datastore to Store Secrets” on page 113.](#)

If you want to do minimal configuration, you can use the configuration datastore on the Administration Console to store the secrets. This option can be used without changes, but is recommended only for use in small Access Manager environments. To increase the security of the secrets, NetIQ recommends that you change the default security options.

---

**IMPORTANT:** Using this option will put additional load on your Administration Console and introduces login delays compared to other options. Therefore it is recommended that this option is used wisely.

---

- ♦ [“Configuring an LDAP Directory to Store the Secrets” on page 114.](#)

This is the recommended option and can be used with any LDAP directory. To use this option, extend the schema to add an attribute to your user object on the LDAP directory that will encrypt and store the secrets.

- ♦ [“Configuring an eDirectory User Store to Use SecretStore” on page 116.](#)

If your user store is eDirectory and you have installed Novell SecretStore, you can choose to use the SecretStore on your eDirectory server to store the secrets. This differs from the schema extension method as Novell SecretStore can also be accessed and managed by Novell SecureLogin. This allows secrets to be shared with SecureLogin to provide a thick client single sign-on while Access Manager can provide a web single sign-on experience without credential collisions.

For some troubleshooting tips, see [“Troubleshooting the Storing of Secrets” on page 118.](#)



## Configuring the Configuration Datastore to Store 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**.

The screenshot shows the 'Credential Profile' configuration page. It has three tabs: 'Details' (selected), 'Descriptions', and 'Custom Attribute Names'. The 'General Settings' section includes a 'Display name' field set to 'Credential Profile' and a checkbox 'Have Discovery Encrypt This Service's Resource Ids'. The 'Credential Profile Settings' section has a checked checkbox 'Allow End Users to See Credential Profile'. The 'Local Storage of Secrets' section includes an 'Encryption Password Hash Key' field with a masked password '\*\*\*\*\*', a 'Preferred Encryption Method' dropdown set to 'Password Based Encryption With MD5 And DES', and an 'Extended Schema User Store References' table with 'New' and 'Delete' buttons and '0 Item(s)'. The table has columns 'User Store' and 'Attribute Name' and shows 'No items'. At the bottom are 'OK', 'Cancel', and 'Apply' buttons.

- 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.

---

**IMPORTANT:** Before using Access Manager to store and encrypt secrets, ensure that you choose your **Preferred Encryption Method** and change the default **Encryption Password Hash Key** value. If either of these options are changed after any secrets are stored, Access Manager will not be able to retrieve the secrets.

---

**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 *NetIQ Access Manager 4.0 SP1 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**.

**Credential Profile**

Edit the details about the web service.

Details | Descriptions | Custom Attribute Names

**General Settings**

Display name:

☐ Have Discovery Encrypt This Service's Resource Ids

**Credential Profile Settings**

☒ Allow End Users to See Credential Profile

**Local Storage of Secrets**

Access Manager controls the storage and encryption of secrets.

Encryption Password Hash Key:

Preferred Encryption Method:

**Extended Schema User Store References**

New | Delete 0 Item(s)

<input type="checkbox"/> User Store	Attribute Name
No items	

OK Cancel Apply

- 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.

---

**IMPORTANT:** Before using Access Manager to store and encrypt secrets, ensure that you choose your **Preferred Encryption Method** and change the default **Encryption Password Hash Key** value. If either of these options are changed after any secrets are stored, Access Manager will not be able to retrieve the secrets.

---

- 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 *NetIQ Access Manager 4.0 SP1 Policy Guide*.

For troubleshooting information, see [“Troubleshooting the Storing of Secrets”](#) on page 118.

## 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 about the rights required, see [Section 3.1.3, “Configuring an Admin User for the User Store,”](#) on page 111.
- ♦ 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.

---

**NOTE:** While configuring new replicas for the same user store, by default the **Use secure LDAP connections** option will be selected and the default port will be 636. The **Use secure LDAP connections** option will be non-editable.

---

- ♦ If you have enabled a firewall between the Administration Console and the user store, and between the Identity Server and the user store, ensure 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 118.

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 SP1 64-bit operating system (or a higher version), the eDirectory server is missing some support libraries that this SAML method requires. For information about installing these libraries, see [TID 7006437 \(http://www.novell.com/support/viewContent.do?externalId=7006437&sliceId=1\)](http://www.novell.com/support/viewContent.do?externalId=7006437&sliceId=1).

4 Click **Liberty > Web Service Providers**.

5 Click **Credential Profile**.

The screenshot shows the 'Credential Profile' configuration window. At the top, there are tabs for 'Details', 'Descriptions', and 'Custom Attribute Names'. The 'Details' tab is selected. Below the tabs, the 'General Settings' section contains a 'Display name' field with the value 'Credential Profile' and a checkbox labeled 'Have Discovery Encrypt This Service's Resource Ids'. The 'Credential Profile Settings' section has a checkbox 'Allow End Users to See Credential Profile' which is checked. The 'Local Storage of Secrets' section includes a note 'Access Manager controls the storage and encryption of secrets.', an 'Encryption Password Hash Key' field with masked characters, and a 'Preferred Encryption Method' dropdown menu set to 'Password Based Encryption With MD5 And DES'. Below this is the 'Extended Schema User Store References' section, which shows a table with columns 'User Store' and 'Attribute Name', and a 'No items' message. At the bottom are 'OK', 'Cancel', and 'Apply' buttons.

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 118](#).
- ♦ To create policies that use the stored secrets, see [“Creating and Managing Shared Secrets” in the \*NetIQ Access Manager 4.0 SP1 Policy Guide\*](#).
- ♦ For troubleshooting information, see [“Troubleshooting the Storing of Secrets” on page 118](#).

## 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 Ensure that 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, ensure 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 *NetIQ Access Manager 4.0 SP1 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

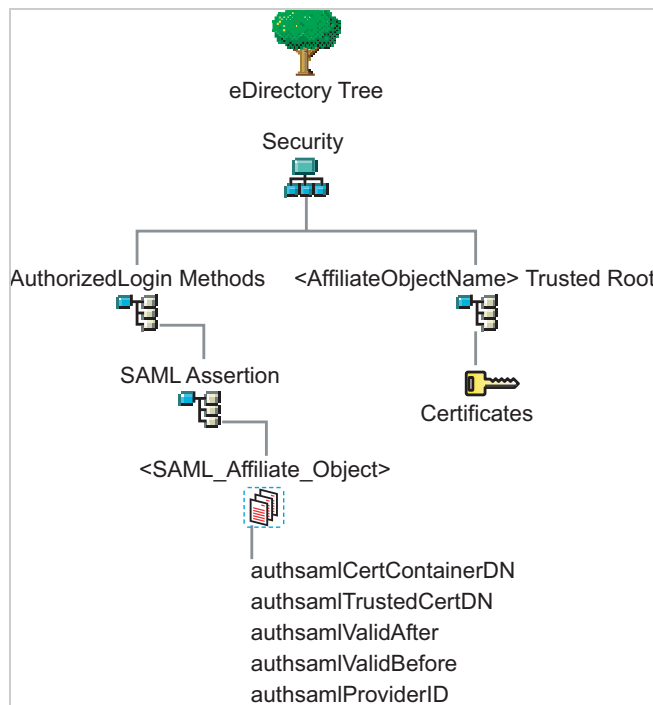
- ♦ [“Secrets Are Not Stored in Novell SecretStore”](#) on page 119
- ♦ [“Users Are Receiving Invalid Credential Messages”](#) on page 120
- ♦ [“Secrets Aren't Stored in the LDAP Directory”](#) on page 120

## Secrets Are Not 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 LDAP 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 [Installation Requirements on Linux](#) in the *NetIQ Access Manager 4.0 SP1 Installation Guide*.
  - ♦ If the objects exist, check for time synchronization problems. For more information, see [“Users Are Receiving Invalid Credential Messages”](#) on page 120.

- 5 In the Administration Console, modify the secret store configuration so that it is resented 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 ensure 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**.



General	Local	Liberty	SAML 1.1	SAML 2.0	WS Federation	Brokering	WS-Trust
User Stores	<b>Classes</b>	Methods	Contracts	Defaults			
New   Delete							
<input type="checkbox"/> <b>Name</b>							
<input type="checkbox"/> <a href="#">Aliasuser</a>							
<input type="checkbox"/> <a href="#">Introductions</a>							
<input type="checkbox"/> <a href="#">Kerberos</a>							
<input type="checkbox"/> <a href="#">Name/Password - Basic</a>							
<input type="checkbox"/> <a href="#">Name/Password - Form</a>							
<input type="checkbox"/> <a href="#">nmas</a>							
<input type="checkbox"/> <a href="#">pwdFetch</a>							
<input type="checkbox"/> <a href="#">Secure Name/Password - Basic</a>							
<input type="checkbox"/> <a href="#">Secure Name/Password - Form</a>							
<input type="checkbox"/> <a href="#">testClass1</a>							
<input type="checkbox"/> <a href="#">Trust Levels</a>							
<input type="checkbox"/> <a href="#">x509</a>							
<input type="button" value="OK"/> <input type="button" value="Cancel"/> <input type="button" value="Apply"/>							

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 introducible identity providers. For information about how to configure and use this class, see [Section 7.2.3, “Configuring the Introductions Class,” on page 208](#).

**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 about how to configure and use this class, see [Section 7.2.4, “Configuring the Trust Levels Class,” on page 209](#).

- 2 To delete a class, select the class, then click **Delete**.

You cannot delete a class if a method is using it.

For information about how to create a name/password class, see the following sections:

- ♦ [Section 3.2.1, “Creating Basic or Form-Based Authentication Classes,” on page 122](#).
- ♦ [Section 3.2.2, “Specifying Common Class Properties,” on page 123](#)

Some classes require additional configuration to enable their use for authentication. See the following sections:

- ♦ [“Configuring for RADIUS Authentication” on page 155](#)

- ♦ “Configuring Mutual SSL (X.509) Authentication” on page 157
- ♦ “Creating an ORed Credential Class” on page 164
- ♦ “Configuring for OpenID Authentication” on page 165
- ♦ “Configuring Password Retrieval” on page 166
- ♦ “Configuring Access Manager for NESCM” on page 169
- ♦ “Configuring for Kerberos Authentication” on page 177
- ♦ “Configuring for Two-Factor Authentication Using Time-Based One-Time Password (TOTP)” on page 150

## 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**.

**Create Authentication Class**

**Step 1 of 2:** Specify name and java class.

Display name:

Java class:

Java class path:

- 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 4.6, “Configuring Mutual SSL \(X.509\) Authentication,” on page 157](#).
    - SocialAuthClass:** The authentication class used for implementing authentication through external OAuth providers such as Facebook, GooglePlus, LinkedIn and Twitter. See [Section 4.1, “Configuring Social Authentication,” on page 145](#).
    - TOTPClass:** The authentication class used for implementing two-factor authentication using Google Authenticator. See [Section 4.2, “Configuring for Two-Factor Authentication Using Time-Based One-Time Password \(TOTP\),” on page 150](#).
    - ProtectedBasicClass:** The BasicClass, protected by HTTPS.
    - ProtectedPasswordClass:** The PasswordClass, protected by HTTPS (form-based).
    - ProtectedRadiusClass:** The RadiusClass, protected by HTTPS. See [Section 4.4, “Configuring for RADIUS Authentication,” on page 155](#) 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 177](#) 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 4.10, “Configuring Access Manager for NESCM,” on page 169](#).

**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 4.7, “Creating an ORed Credential Class,” on page 164](#).

**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 4.9, “Configuring Password Retrieval,” on page 166](#).

**PersistentAuthClass:** The authentication class that allows for persistent logins, long authentication sessions, or remember my password functionality. For configuration information, see [Section 4.3, “Configuring Persistent Authentication,” on page 153](#).

**Other:** Used for third-party authentication classes or if you have written your own Java class. For information about how to write your own class, see [Novell Access Manager Developer Tools and Examples \(http://www.novell.com/developer/ndk/novell\\_access\\_manager\\_developer\\_tools\\_and\\_examples.html\)](#).

**AliasUserPasswordClass:** This class supports authentication of a user against user’s alias name. This class uses the alias object of the user object and the password of the corresponding user object to authenticate.

- 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 123](#) 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 125](#).

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 124](#)
- ♦ [“JSP Property” on page 124](#)
- ♦ [“MainJSP Property” on page 125](#)

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 about creating a custom login page, see [Section 2.1, "Customizing the Identity Server Login Page," on page 61](#).

## 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 about how to create a custom login page, see [Section 2.1, “Customizing the Identity Server Login Page,” on page 61](#).

## 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 about how to create a custom login page, see [Section 2.1, “Customizing the Identity Server Login Page,” on page 61](#).

## 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**.

General			Local	Liberty	SAML 1.1	SAML 2.0	WS Federation	Brokering	WS-Trust
User Stores   Classes   <b>Methods</b>   Contracts   Defaults									
New   Delete									
<input type="checkbox"/> Name	Identifies User	Class							
<input type="checkbox"/> <a href="#">aliasuser</a>	<input checked="" type="checkbox"/>	Aliasuser							
<input type="checkbox"/> <a href="#">formLogin</a>	<input checked="" type="checkbox"/>	Name/Password - Form							
<input type="checkbox"/> <a href="#">kerberos</a>	<input checked="" type="checkbox"/>	Kerberos							
<input type="checkbox"/> <a href="#">Name/Password - Basic</a>	<input checked="" type="checkbox"/>	Name/Password - Basic							
<input type="checkbox"/> <a href="#">Name/Password - Form</a>	<input checked="" type="checkbox"/>	Name/Password - Form							
<input type="checkbox"/> <a href="#">nmas</a>	<input checked="" type="checkbox"/>	nmas							
<input type="checkbox"/> <a href="#">pwdFetch</a>	<input checked="" type="checkbox"/>	pwdFetch							
<input type="checkbox"/> <a href="#">Secure Name/Password - Basic</a>	<input checked="" type="checkbox"/>	Secure Name/Password - Basic							
<input type="checkbox"/> <a href="#">Secure Name/Password - Form</a>	<input checked="" type="checkbox"/>	Secure Name/Password - Form							
<div>OK Cancel Apply</div>									

- 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**.

formLogin

Display name:

formLogin

Class:

Name/Password - Form

☒ Identifies User
 ☒ Overwrite Temporary User
 ☒ Overwrite Real User

User stores:

<Default User Store>

Available user stores:

idp205

local

↑

↓

Properties

New | Delete

☐ Name Value

No items

OK Cancel Apply

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 120](#).

**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.

To achieve SSO on backend Web application when the passwordfetch class is enabled, see [TID](#).

**Overwrite Temporary User:** If you select this check box, then the temporary user credentials profile got from previous authentication method in the same session will be overwritten with real user credentials profile got from this authentication method.

**Overwrite Real User:** If you select this check box, then the real user credentials profile got from previous authentication method in the same session will be overwritten with real user credentials profile got from this authentication method.

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 137](#).

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 about the available properties for the basic and form classes, see [Section 3.2.2, “Specifying Common Class Properties,” on page 123](#)

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.

If this method is part of a multi-factor authentication, you can set the following additional property:

- ♦ **PRINCIPAL\_MISMATCH\_ERR**: Specifies the error message to be displayed if this method identifies a different principal than other methods in the multi-factor authentication.

**Property Value:** The values associated with the **Property Name** field.

7 Click **Finish**.

8 Continue with [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).

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**.

General			Local	Liberty	SAML 1.1	SAML 2.0	WS Federation	Brokering	WS-Trust
User Stores			Classes	Methods	Contracts	Defaults			
New			Delete						
<input type="checkbox"/>	Name	URI				Level			
<input type="checkbox"/>	<a href="#">aliasuser</a>	/name/pwd/alias				0			
<input type="checkbox"/>	<a href="#">idp_205</a>	name/password1/uri				0			
<input type="checkbox"/>	<a href="#">kerberos</a>	/kerb/uri				0			
<input type="checkbox"/>	<a href="#">MultipleMethods</a>	Multiple/methods				0			
<input type="checkbox"/>	<a href="#">Name/Password - Basic</a>	basic/name/password/uri				0			
<input type="checkbox"/>	<a href="#">Name/Password - Form</a>	name/password/uri				0			
<input type="checkbox"/>	<a href="#">NMA5</a>	/nmas/uri				0			
<input type="checkbox"/>	<a href="#">Secure Name/Password - Basic</a>	secure/basic/name/password/uri				0			
<input type="checkbox"/>	<a href="#">Secure Name/Password - Form</a>	secure/name/password/uri				0			
<input type="checkbox"/>	<a href="#">testForm</a>	/test/form				0			
OK			Cancel			Apply			

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.

3 To create a new contract, click **New**.



**Configuration Authentication Card**

Display name:

URI:

Password expiration servlet:

☐ Allow user interaction

Login Redirect URL:

☒ Allow user interaction

Authentication Level:

Authentication Timeout:  Minutes

Activity Realm(s):

☐ Satisfiable by a contract of equal or higher level

☒ Satisfiable by External Provider

Requested By:

Allowable Class:

If you add more than one X509 method, only the first one will be used and it will automatically be moved to the top of the list.

Methods:

Available methods:

#### 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 password when the password expires or is within the grace login period. You must use eDirectory to change the number of grace logins. Grace logins work only with eDirectory.

For more information about how to use this type of servlet, see [Section 3.4.1, "Using a Password Expiration Service," on page 132](#).

**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.

**Login Redirect URL:** You will be redirected to the URL specified in this field. Use this field for the following scenarios:

- ♦ Forcing the user to a specific home page after successful Access Manager authentication.
- ♦ Forcing the user to configure challenge/response forgotten password questions

For more information about the URL parameters, see [Section 3.4.2, "Using Login Redirect URL Parameters," on page 135](#).

**Allow User Interaction:** You can enable this option, which allows the user to decide whether to continue to access a pre-configured URL or to continue to the page that the user usually accesses. For example, the user may usually access `www.a.com` and have specified the redirect URL as `https://someservice.com/path/password?user=<USERID>&store=<STOREID>&returl=<RETURN_URL>` then, continue will allow you to continue with the website you access i.e. `www.a.com` and redirect URL will take you to the URL `https://someservice.com/path/password?user=<USERID>&store=<STOREID>&returl=<RETURN_URL>&action=expire` and then to `www.a.com`.

**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.

---

**NOTE:** In case of **Name/Password - Basic** and **Secure Name/Password - Basic** contracts applied to a protected resource, then you won't find the session as timed out, as the session gets renewed after timeout without user intervention using the Basic header sent from browser to Identity Provider.

---

For information about how to use this feature with the Access Gateway, see [“Assigning a Timeout Per Protected Resource”](#) in the *NetIQ Access Manager 4.0 SP1 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 3.4.3, “Admin can configure some more parameters that wiUsing Activity Realms,”](#) on page 135.

**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.

When the protected resource is configured with **Name/Password -Form** as Authentication procedure, the user authentication details are prompted with transient federation. This option should be enabled to avoid prompting for authentication in the Target Service Provider.

**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.

**Requested By:** Select one of the following options:

- ♦ **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 128](#).

Most third-party identity providers do not use contracts.

**Allowable Class:** Specifies the class that instructs a service provider to send a request for a specific authentication type to the Identity Provider. You are allowed to modify this option only when you select authentication types.

---

**NOTE:** In SAML 2 federation with Access Manager as Service Provider, if external Identity Server is authenticating a user, it sends `<AuthnContext>` information after authentication in the response. Access Manager uses this `<AuthnContext>` to find a matching contract at the Service Provider to identify the user. It identifies the contract by trying to match

`<saml:AuthnContextClassRef>` with AllowableClass attribute or

`<saml:AuthnContextDeclRef>` with URI attribute of existing contracts at the Service Provider.

---

For example, if the external Identity Server sends the following AuthnContext

```
<saml:AuthnContext>
```

```
<saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</saml:AuthnContextClassRef>
```

```
<saml:AuthnContextDeclRef>adroit:login:user:np</saml:AuthnContextDeclRef>
```

```
</saml:AuthnContext>
```

and if Access Manager (as a Service Provider) has a contract A with uri = adroit:login:user:np, or with Allowable class =

urn:oasis:names:tc:SAML:2.0:ac:classes:Password, then it matches the contract.

---

**NOTE:** The Allowable class field is blank when an inbuilt Authentication Class is used in Identity Server.

---

For more information about using CloudAccess as a trusted Identity Provider, see [Using NetIQ® CloudAccess as a Trusted Identity Provider for NetIQ® Access Manager](#).

**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.

You can enable the multi-factor authentication by associating more than one methods to a contract.

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 [“Enabling SSL Communication”](#) in the *NetIQ Access Manager 4.0 SP1 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 137](#).
- ♦ You can configure a protected resource to use it. See [“Configuring Protected Resources”](#) in the *NetIQ Access Manager 4.0 SP1 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](#).

As you configure the service, be aware of the following configuration options:

- ♦ [“URL Parameters” on page 133](#)
- ♦ [“Forcing Authentication after the Password Has Changed” on page 133](#)
- ♦ [“Grace Logins” on page 134](#)

- ♦ [“Federated Accounts” on page 134](#)
- ♦ [“Redirection to Password Management Servlet Protected by Access Gateway When Password Expires” on page 134](#)

## 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, ensure that 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 \(http://www.novell.com/support/php/search.do?cmd=displayKC&docType=kc&externalId=3465171&sliceId=2&docTypeID=DT\\_TID\\_1\\_1&dialogID=131458644&statId=0%200%20131454892\)](http://www.novell.com/support/php/search.do?cmd=displayKC&docType=kc&externalId=3465171&sliceId=2&docTypeID=DT_TID_1_1&dialogID=131458644&statId=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, or WS Federation 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.

## Redirection to Password Management Servlet Protected by Access Gateway When Password Expires

When an Active Directory user with an expired password logs in to an authentication contract with a Password Expiration servlet configured, the user is redirected to the password management URI. If the Password Management portal is protected by Access Manager, the user is prompted again for authentication and is not permitted to login as the user password has expired.

If you want the user to be redirected to the Password Management Servlet, perform the following steps:

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > Local > Methods**.
- 2 Select the authentication method, which is used by the contract where Password Management Servlet is configured.
- 3 Add the following property for the method used by contract with Password Expiration servlet:  
ExpiredCheck=true
- 4 Add the following property for the method used by contract that protects the Password Management portal:  
ExpiredCheck=true ExpireCheck=true

- 5 Open the `/opt/novell/nam/idp/webapps/nidp/WEB-INF/classes/nidpconfig.properties` file.  
Add the `AUTHENTICATE_WITH_EXPIRED_PASSWORD` property to the file.  
For example: `AUTHENTICATE_WITH_EXPIRED_PASSWORD=ad/name/password/uri`  
Repeat this step for Identity Server cluster members.
- 6 Click **OK**, **Apply**, and then **Update** the Identity Server.

## 3.4.2 Using Login Redirect URL Parameters

When you are defining the URL for login redirect URL 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 login redirect URL must understand the name-value pairs you have defined and will use the resolved values in the redirected URL.

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.

For example:

```
https://someservice.com/path/password?user=<USERID>&store=<STOREID>
&returl=<RETURN_URL>
```

**NOTE:** If you copy and paste this text, ensure that you remove the white space between `<STOREID>` and `&returl`.

The Identity Server fills in these values, that results in the following URL:

```
https://someservice.com/path/password?user=joe.novell&store=userstore1&
returl=https://myidp.com/nidp/idff/sso
```

In addition to the above three parameters you can also configure other parameters

## 3.4.3 Admin can configure some more parameters that wiUsing Activity Realms

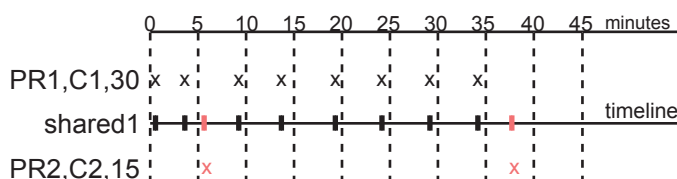
Activity realms are designed to be used with an Access Manager system that uses multiple contracts to protect resources that require different activity timeouts. Activity realms allow you to define how activity at one protected resource affects the activity timeout at another protected resource.

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 timeout 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 *NetIQ Access Manager 4.0 SP1 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**.

The screenshot shows the 'idp-corporate' configuration page. At the top, there are tabs for 'General', 'Local', 'Liberty', 'SAML 1.1', 'SAML 2.0', 'WS Federation', and 'Bro'. Below these tabs is a sub-menu with 'User Stores', 'Classes', 'Methods', 'Contracts', and 'Defaults'. The 'Defaults' section is active and contains the following fields:

- User Store: Internal (dropdown menu)
- Authentication Contract: Name/Password - Form (dropdown menu)

Below these fields is a table with two columns: 'Authentication Type' and 'Default Contract'.

Authentication Type	Default Contract
Name Password:	<None> (dropdown menu)
Secure Name Password:	<None> (dropdown menu)
X509:	<None> (dropdown menu)
Smart Card:	<None> (dropdown menu)
Smart Card PKI:	<None> (dropdown menu)
Token:	<None> (dropdown menu)

- 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 [“NetIQ Access Manager 4.0 SP1 Access Gateway Guide”](#) in the *NetIQ Access Manager 4.0 SP1 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 138](#).

- 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 138](#).

- 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 209](#).

## 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 128](#).

- 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 128](#).

- 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.

- [Section 3.6.1, "Logging In to User Portal," on page 139](#)
- [Section 3.6.2, "Specifying a Target," on page 140](#)
- [Section 3.6.3, "Blocking Access to the User Portal Page," on page 140](#)
- [Section 3.6.4, "Blocking Access to the WSDL Services Page," on page 142](#)

### 3.6.1 Logging In to 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.

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, ensure that 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 NetIQ Web site after logging in.

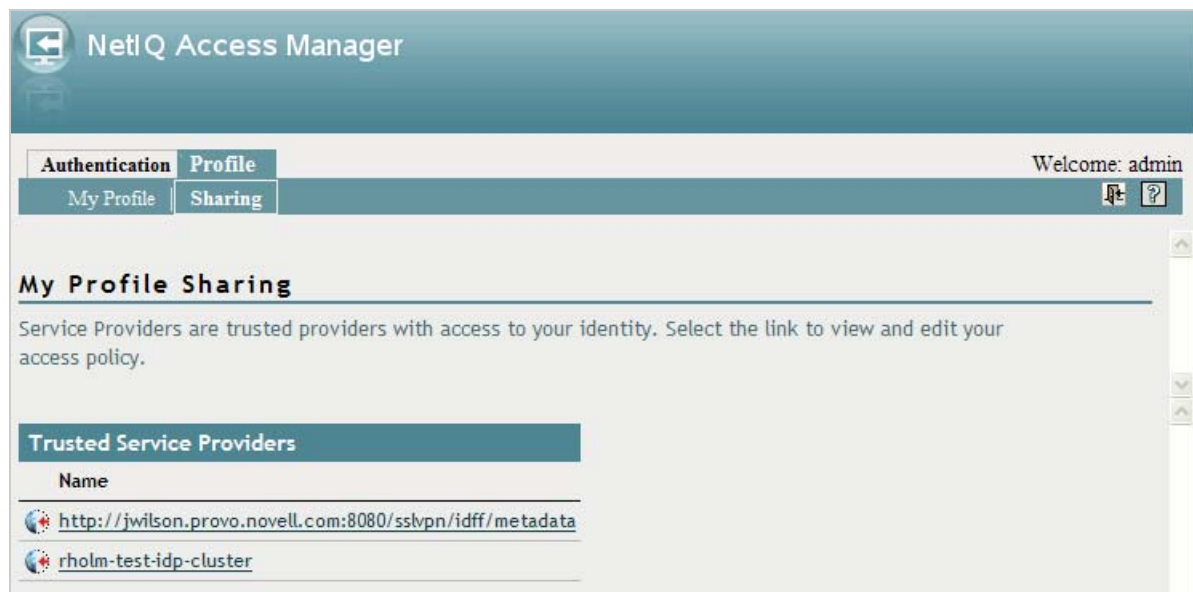
- **Specify a Hidden Target on your Form:** If you have your own login form to collect credentials and are posting these credentials to the Identity Server, you can add a hidden target to your login form. When authentication succeeds, the user is directed to this target URL. This entry on your form should 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:** `/opt/novell/nids/lib/webapp/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: idpl26.lab.novell.com:8443
Connection: Keep-Alive
Cookie: JSESSIONID=11AB34250B3E79DEC11186168C23B34D; novell_language=en-us;
CoreID6=23495995982212440449949;
__utma=64695856.419410920.1252432782.1270822885.1271090179.10;
__utmz=64695856.1270722077.8.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none);
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.

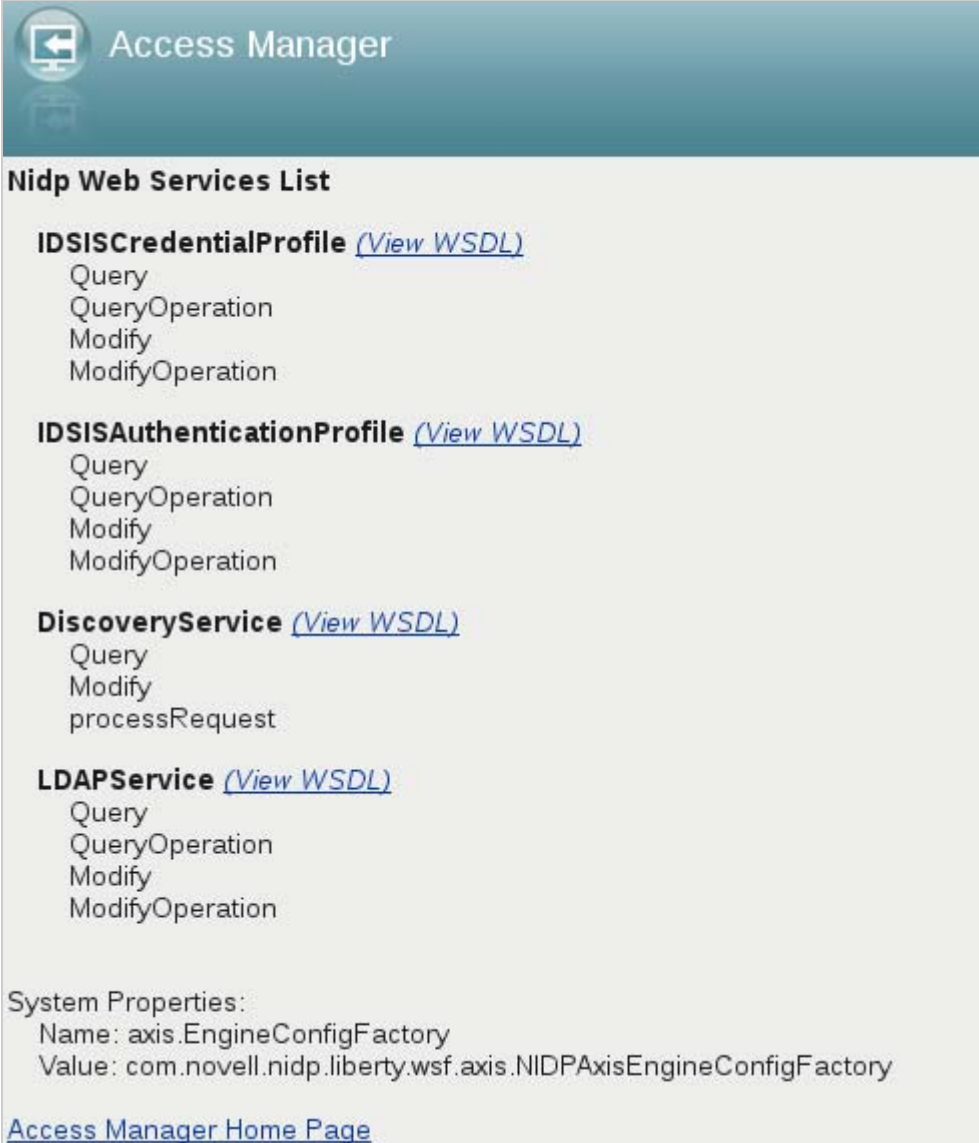
### 3.6.4 Blocking Access to the WSDL Services Page

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 3-4** WSDL Services Page

The screenshot shows the 'Access Manager' interface with a 'Nidp Web Services List'. It contains four service entries: 'IDISCredentialProfile', 'IDISAuthenticationProfile', 'DiscoveryService', and 'LDAPService'. Each entry has a '(View WSDL)' link and a list of operations. At the bottom, there are 'System Properties' and a link to the 'Access Manager Home 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 from this location:  
**Linux:** `/opt/novell/nids/lib/webapp/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 one of the following commands:

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

```
rcnovell-idp restart
```

**Windows:** Enter the following commands:

```
net stop Tomcat7
```

```
net start Tomcat7
```



---

# 4 Configuring Advanced Local Authentication Procedures

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.

- ♦ [Section 4.1, “Configuring Social Authentication,” on page 145](#)
- ♦ [Section 4.2, “Configuring for Two-Factor Authentication Using Time-Based One-Time Password \(TOTP\),” on page 150](#)
- ♦ [Section 4.3, “Configuring Persistent Authentication,” on page 153](#)
- ♦ [Section 4.4, “Configuring for RADIUS Authentication,” on page 155](#)
- ♦ [Section 4.5, “Configuring Client Integrity Check,” on page 156](#)
- ♦ [Section 4.6, “Configuring Mutual SSL \(X.509\) Authentication,” on page 157](#)
- ♦ [Section 4.7, “Creating an ORed Credential Class,” on page 164](#)
- ♦ [Section 4.8, “Configuring for OpenID Authentication,” on page 165](#)
- ♦ [Section 4.9, “Configuring Password Retrieval,” on page 166](#)
- ♦ [Section 4.10, “Configuring Access Manager for NESCM,” on page 169](#)

## 4.1 Configuring Social Authentication

Access Manager can be configured to support authentication through external OAuth providers like Facebook, Google+, Twitter, LinkedIn, and so on. Social authentication simplifies login for end users and does not require maintaining large user stores. This authentication can be configured using the SocialAuthClass. Login using social identities provide a convenient way for users, improving customer satisfaction and increased registration levels. For more information on how to configure the supported social authentication providers for API Keys and API Secrets, see [Appendix D, “Configuring the Supported Social Authentication Providers for API Keys and API Secrets,” on page 493](#).

Social login allows business, universities and government entities to leverage social identity providers to share select identity information for authentication via OAuth tokens. This information can then be used to provide protected online services ranging from customer-focused applications, university sites to state and local services and more.

### 4.1.1 Use Case

Authentication through external OAuth providers can be useful in the following two scenarios:

- ♦ Allow external users to access secure resource

For example, you may want your customers and partners to access <https://forums.novell.com>. Creating and managing these external users is a hassle for you and the user. Social Authentication helps in this scenario.

Users will be allowed to sign in with their Facebook or Yahoo ID. Social authentication provider will give Access Manager a set of logged-in user's attributes. Hence, you will get user data without maintaining it. Access Manager can use this user data and perform actions based on that if required.

- ♦ Apply policies to restrict users to access a protected resource

If the **Identify User Locally** option is selected, the social provider user will be mapped to the local user and you can execute authorization policies based on the user attributes. For example, if Joe is a Facebook user, you can match the attributes of Joe in the local user store based on a rule and execute an authorization policy to access a protected resource. You want to apply policies on an incoming user. For example, your enterprise user 'Bob' has logged into <https://forums.novell.com/> with a social identity. You may want to identify that 'Bob' is your local user and provide him with forum moderator privileges. The **Identify User Locally** option lets you map a social user to your local user and apply appropriate policies.

- ♦ **Simplify user login:** You may want to keep the user in your user stores but still make the registration process easy for the users. Social authentication saves the user from remembering another identity. User can login with their social identity while the **Auto Provision User** option will map the incoming user specified attribute with an existing user in the local user store. If the attribute matches, user will be provisioned, else user will be prompted for local user authentication.
- ♦ **Personalized web content in B2C scenarios:** Organizations want to make services and information available in a manner that is personalized to individual. The common approach of creating individual identities for users is costly for the organization and inconvenient for the user. Social login allow users to login with their preferred form of identities. This simplifies the login experience for customers while increasing the registration levels and lowering IT costs.
- ♦ **Step up authentication:** While you as an administrator want to improve the user registration through social identities, you would also want to ensure that a second factor authentication is employed when users access sensitive information. Access Manager provides options to configure multiple contracts for protected resources and as users access these resources, they can be prompted to login with a second factor such as their corporate identity or an OTP.

## 4.1.2 Prerequisite

You must have registered Access Manager with the social authentication providers and should have the API keys and API secrets for establishing federation between Access Manager and the provider for example, Facebook.

## 4.1.3 Configuring SocialAuthClass

Use the Administration Console to define a new Social Authenticator class, method and contract for the Identity Server cluster. Social authenticator providers such as Facebook, Google+, LinkedIn and Twitter are supported.

- 1 Login to the Administration Console.
- 2 Click **Devices > Identity Servers > Edit > Local > Classes**. Select **New** to add a new class.
- 3 Specify a name to identify the class. For example, Social authenticator.
- 4 Select **SocialAuthClass** from the **Java Class** drop-down list. Click **Next**.

5 Configure the **User Identification** settings if you need to perform actions on the logged in user. This is optional. By default, user authentication is done without mapping the social provider user to a local user.

- ♦ **Identify User Locally:** Select this option to map the incoming user to an existing user in your user store. You can apply an authorization policy for these incoming users to provide access control. The following two parameters specify how to perform the user mapping:

- ♦ **Local User LDAP Attribute:** Select an attribute from the drop-down list, for example **LDAP Attribute:mail [LDAP Attribute Profile]**. The incoming configured attribute from the social website is mapped to local user's LDAP attribute.

---

**NOTE:** If there are more than one social authentication providers configured, the **Local User LDAP** attribute must be a multi-valued attribute. This is required to store the social attributes corresponding to each social provider.

---

- ♦ **Social User Attribute:** Select an attribute which provides a unique user identity for example **Email**. The user email provided in a social website will be mapped to the specified local user's LDAP attribute.

User mapping is done if the value of **Local User Attribute** is equal to the value of **Social User Attribute**.

---

**NOTE:** Provisioning will not occur in the following scenarios:

- ♦ If you are going to use Facebook or Google+ as your authentication provider, do not select **DisplayName** as **Social User Attribute** as these providers do not have the **DisplayName** attribute.
  - ♦ If Social User Attribute is email attribute in Twitter.
- 

- ♦ **Auto Provision User:** If you enable this option, incoming user specified attribute will be mapped with an existing user in the local user store. If the attribute matches, user will be provisioned, else user will be prompted for local user authentication. After authentication, user attribute will be mapped and stored.

6 Click **Add** under **Social Auth Providers** to provide the authentication provider details.

- ♦ **Auth Provider:** Select the authentication provider from the drop-down list for example, Facebook. You can select from one of the predefined providers or select **Other** to specify your own providers. Note that only the predefined providers have been verified for compatibility with Access Manager. If you select **Other**, you must provide two additional information:
  - ♦ **Provider Name:** Specify the name of the provider. Other provider names can be specified under **Others** option. Other provider name can be Yahoo, Hotmail, Salesforce, AOL, FourSquare, MySpace, Instagram, Mendeley or Yammer. Name of social authentication provider is case-sensitive and must match as listed. Else, social authentication class will not work.
  - ♦ (Optional) **Implementation Class:** Specify a back end class that can authenticate with these providers if the other providers are not supported. This is needed only for a custom provider that is not in the list provided above.
- ♦ **Consumer Key:** Specify the API key that you received when you registered Access Manager with the Social authentication provider.
- ♦ **Consumer Secret:** Specify the secret that you received when you registered Access Manager with the Social authentication provider.

7 Click **OK** and **Finish**.

- 8 Continue with creating a contract and method for this class.

For configuration information, see [Section 3.3, “Configuring Authentication Methods,” on page 125](#) and [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).

## How Social Authentication Works With Access Manager

For completing social authentication, the Identity Server maps the social attribute value in token to the local user attribute value. The local attribute must be set in the following format for the matching to succeed.

```
<socialprovidername>:<social attribute value>
```

For example, consider that the social authentication class properties are set as follows:

- ♦ **Identify User Locally:** Enabled
- ♦ **Local User LDAP attribute:** Ldap Attribute:mail
- ♦ **Social User Attribute:** Email
- ♦ **Auto Provision User:** Enabled
- ♦ **Social Auth Provider:** Facebook

As the **Auto Provision User** setting is enabled, after authentication in Facebook, user is asked for a one-time local login. During this process, this user's mail attribute is updated with the social attribute value as `facebook:<social-email-address>`. Subsequent logins from the same user will be seamless and user will be identified automatically.

If **Auto Provision User** setting is disabled, for the authentication to succeed, Access Manager will check if local user LDAP attribute mail value is `facebook:<social-email-address>`.

---

**NOTE:** The attribute value is set with the provider's name.

---

### 4.1.4 Adding Images for Social Authentication Providers

You can add images for social authentication providers such as Facebook, LinkedIn, Twitter, Google+ and so on. For more information about adding images, see [Section 6.5, “Adding Authentication Card Images,” on page 199](#).

- 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 199](#).

- 6 Add all the required images and click **Close**.

After configuring the Identity Server with the required social authentication provider images, the login page will display those images as in [Figure 4-1](#). The User Login screen will ask you to choose and access the social providers you have added when you access the Identity Server URL.

**Figure 4-1** User Login Screen with Social Authentication Provider Images



## 4.1.5 Changing the Social Authentication Icons

The following procedure allows you to change the default icons of social authentication providers.

- 1 Go to `socialauth_icons.jsp` file located at `/opt/novell/nids/lib/webapp/jsp/`. You can see all the supported providers and their corresponding public URL locations.
- 2 To change the icon of a particular provider, go to the icon variable name of that provider and replace the existing URL location with required URL location.  
You can similarly change for other icons defined in the jsp file.
- 3 Restart the Identity Server after changing the jsp file.

## 4.1.6 Configuring the Supported Social Authentication Providers for API Keys and API Secrets

Access Manager requires API Keys and API Secrets from the supported social authentication providers to integrate with these providers. For more information about configuring the supported applications, see [Appendix D, "Configuring the Supported Social Authentication Providers for API Keys and API Secrets,"](#) on page 493.

## 4.2 Configuring for Two-Factor Authentication Using Time-Based One-Time Password (TOTP)

This section explains how to use Google Authenticator Time-Based One-Time Password (TOTP) as a second authentication factor with Access Manager. The Google Authenticator uses a six-digit number (OTP) in addition to first authentication (for example: username, password), to log into protected services.

The first step is to register the Google Authenticator client with the secret key. This secret key is used for all future logins to the Web Site.

Typically, users download and install the Google Authenticator app on their devices. To log into a Web Site or service that uses two-factor authentication, in addition to the user name and password, the users enter a additional OTP generated by the Google Authenticator app. Access Manager validates the OTP and authenticates the user.

For more information about implementation, see [Google Authenticator on Wikipedia](#)

- ♦ [Section 4.2.1, “Why Two-Factor Authentication?,” on page 150](#)
- ♦ [Section 4.2.2, “Prerequisite,” on page 150](#)
- ♦ [Section 4.2.3, “Configuring TOTP Class, Method, and Contract,” on page 150](#)
- ♦ [Section 4.2.4, “Registering with Google Authenticator,” on page 152](#)
- ♦ [Section 4.2.5, “Verifying TOTP Configuration,” on page 152](#)

### 4.2.1 Why Two-Factor Authentication?

Two-factor authentication such as TOTP provides additional security for the systems. It works on the principle of granting access based on a knowledge factor (something the user has) and a possession factor (something the user knows). This helps organizations that need to implement a multi-factor authentication scheme to satisfy regulatory requirements or increase security.

### 4.2.2 Prerequisite

- ♦ Download and install the Google Authenticator app on your device. This app generates an OTP that is later used for authentication.
- ♦ Google Authenticator relies on the device time (of the Google Authenticator app) to generate an OTP. So, it is important that the time on your device is accurate.

### 4.2.3 Configuring TOTP Class, Method, and Contract

Use the Administration Console to define a new TOTP Authenticator class, method, and contract for the Identity Server cluster.

- 1 Log in to the Administration Console.
- 2 Click **Devices > Identity Servers > Edit > Local > Classes > News** to add a new class.
- 3 Specify a name to identify the class. For example, Google authenticator.
- 4 Select TOTPClass from the **Java Class** option. The **Java class path** is displayed as `com.novell.nidp.authentication.local.TOTPAuthenticationClass`. Click **Apply** to save the changes. By default, the TOTP class stores the secret key in the Shared Secret store and no further configuration is required.

- 5 [Optional] You can also optionally store the secret key in an LDAP attribute, file or memory. To do that follow the steps outlined in this table:

---

**NOTE:** File and Memory class implementation are not recommended for production deployment and are only suitable for a single node Identity Server test environment.

---

**LDAP user attribute:** This option stores the secret key on an LDAP attribute of the user object in the user store.

1. Select the **Properties** tab of the Google Authenticator class configuration. Add a new property to indicate that the secret key should be stored in an LDAP attribute of the user object in the user store.

Specify the **Property Name** as `SECRET_STORE_CLASS` and **Property Value** as `USERSTORE`.

2. Add another property to indicate the attribute in which the secret key should be stored.

Specify the **Property Name** as `SECRET_LDAP_ATTRIBUTE_NAME` and specify the name of any single-valued attribute. For example, you can specify the **Property Value** as `mobile`, `costcentre` etc.

The secret key is encrypted and stored in the LDAP attribute. If you do not specify any **Property Value**, the secret key is stored in the `carLicense` LDAP attribute.

---

**NOTE:** Do not use a multi-valued LDAP attribute like `email address` as the **Property Value** as the user registration will fail. It is also important to ensure that the LDAP attribute you have specified as the **Property Value** is a non-operational attribute. For example, it is not recommended to use LDAP Attributes like `groupmembership`.

---

**File class:** The File class writes the secret key to a file on the Identity Server file system.

Select the **Properties** tab of the Google Authenticator class configuration and add a new property to have the user's secret key stored in a file on the file system.

Specify the **Property Name** as `SECRET_STORE_CLASS` and **Property Value** as `FILE`.

**Memory class:** The Memory based class writes the secret key into memory. This memory is transient in nature and therefore the secret key value is lost each time the Identity Server is restarted.

Select the **Properties** tab of the Google Authenticator class configuration and add a new property to define the memory-based property where each user's secret key is stored.

Specify the **Property Name** as `SECRET_STORE_CLASS` and **Property Value** as `MEMORY`.

- 6 Click **Devices > Identity Servers > Edit > Local > Methods**. Select **New** to add a new method.
- 7 Specify a name to identify the method. Select the Google authenticator class from drop-down list. This links the Google authenticator class to the authentication method.
- 8 Deselect the **Identifies User** option. Click **Apply** to save the changes.
- 9 Select the user store from the **list of Available user stores** and move it to **User store**.
- 10 You can either use an existing authentication contract or create a new authentication contract. For example, you can add the default `Name/Password - Form` method as the first method and Google Authenticator method as the second method. Click **Apply** to save the changes.

---

**NOTE:** If you use TOTP as a post-authentication method in a federation setup, a `JSP file not found` message is displayed and federation fails.

---

## 4.2.4 Registering with Google Authenticator

- 1 Go to NetIQ Identity Server page `http(s)://<idp server>:<port>/nidp`
- 2 Select the contract where Google Authenticator is configured as the second method for two-factor authentication.
- 3 Login with the first method.
- 4 Click the link beside **Please register for two factor authentication** to generate a OTP. Make a note of the secret key displayed.

If you have installed the Google Authenticator client on your device, scan the code. You can also manually enter the secret key in the Google Authenticator mobile client.



After the registration is complete on the Google Authenticator client on your mobile, the OTP is displayed.

## 4.2.5 Verifying TOTP Configuration

- 1 Go to NetIQ Identity Server page: `http(s)://<idp server>:<port>/nidp`
- 2 Select the contract where Google Authenticator is configured as the second method for two-factor authentication.
- 3 Login with the first method.

After successfully authenticating with the username and password, prompt is displayed to enter the Google Authenticator OTP.

- 4 Use the Google Authenticator app to generate the OTP and login using this OTP.



## 4.3 Configuring Persistent Authentication

This authentication class stores user session on the browser after successful login. When the user is prompted for authentication subsequently, this class will reuse the saved authentication instead of prompting the user for credentials. The user will be prompted for credentials again only when the cookie lifetime expires. This authentication class is used only for applications that do not require very high security.

### 4.3.1 Frequent Re-authentication Using Password

This class helps in configuring websites that have low security such as enterprise forums. Frequently typing the password to re-authenticate may be vulnerable and cause security issues. To avoid this with `PersistentAuthClass` configuration you will not be required to re-authenticate using the password frequently. For sites that you use a low-grade identity for example, enterprise forums or some web sites that restrain your preferences, having to re-authenticate every few-hours is annoying. Some web sites offer the remember my password feature that will not ask the user to re-authenticate if you select this option. This class provides that remember my password functionality so that the user does not have to frequently re-authenticate.

### 4.3.2 PersistentAuthClass Properties

You can set the following class properties in the configuration file.

- ♦ **CryptoKey**: This key is used to encrypt the user's information in the cookie. If this value is long and random, the user information will be secure. The value must be at least ten characters. The certificate private key will be used if you do not set this value. If you change or update the certificate, the user is re-authenticated.
- ♦ **CookieSuffix**: The Cookie Name is derived using this suffix. PA\_ is added as a prefix to the cookie name. By default, cookie name is PA\_PERSISTENT\_AUTH. For example, if you configure the CookieSuffix as PER\_AUTH, the Identity server sends cookie with PA\_PER\_AUTH name at browser.
- ♦ **MaxAgeSeconds**: This property will decide the cookie lifetime. Default value is 86400 seconds (1 day). Maximum value is 4294967295 seconds.
- ♦ **ParamName**: The name of the HTTP parameter to enable this feature. The default value of the parameter is EnableCookieAuth. If you want to modify the default value of parameter name for example to TestCookieName, follow the procedure given below.

1. Login to the Identity Server.
2. Go to `/opt/novell/nids/lib/webapp/jsp`
3. Open `login.jsp` file using an editor.
4. Search for `EnableCookieAuth` parameter name and provide the new name as `TestCookieName` in the input tag.
5. Ensure that you select the **Remember Me** option.
6. Restart the Identity Server.

This value is used by the Identity Server to identify if user has enabled **Remember Me** option on the login page.

### 4.3.3 Configuring Persistent Authenticator Class

The following procedure allows you to configure the PersistentAuthClass.

- 1 Login to the Administration Console.
- 2 Click **Devices > Identity Servers > Edit > Local > Classes**.
- 3 Click **New**, then specify a **Display name** for example, PersistentAuth.
- 4 Select **PersistentAuthClass** from the **Java Class** drop-down list.
- 5 Click **New** to create a new authentication class.
- 6 In the Add property window, specify the following values. Specifying these values are optional.
  - ♦ **Property Name:** Specify the name of the property. For more information on the names you can specify here, see [Section 4.3.2, “PersistentAuthClass Properties,” on page 153](#)
  - ♦ **Property Value:** Specify the property value you would like to define here.
- 7 Click **OK** and **Finish**.
- 8 Continue with creating a contract and method for this class.

For configuration information, see [Section 3.3, “Configuring Authentication Methods,” on page 125](#) and [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).

### 4.3.4 Logging Out of the Persistent Sessions

When a user performs an explicit logout, the Identity Server clears the persistent authentication cookie at browser if the logout request goes through the browser. If SOAP communication is used between the service provider and the Identity server, then the Identity server does not clear the cookie automatically. The cookie can only be cleared by sending a request to a page on the server that issued it. If the page is available on the Identity server, the `clearCookieAuth.jsp` file clears the page. You must customize the service provider's logout page to run the Identity server's `clearCookieAuth.jsp` page.

The `clearCookieAuth.jsp` file clears it. The URL for this page will be <https://idpserver.example.com/nidp/clearCookieAuth.jsp>. Any request to that URL will clear the authentication cookie.

With this class in use, the user will be unable to logout of the system because re-accessing any protected page will simply re-authenticate the user using the user information stored in the browser stored. There are at least two ways to invalidate an outstanding browser stored authentication cookie. The first is to change the user's password and second is to clear the stored cookie from the browser. Only way to invalidate the cookie is to change the encryption key used. The cookie that is created can only be cleared by a request from the server which created it.

The following configurations are specific to the Novell service provider. If the users uses third party service provider, then the user must customize the logout pages.

In a federation scenario add the following to the `logoutSuccess.jsp` file at `/opt/novell/nam/idp/webapps/nidp/jsp/` of the service provider. You can have logout page redirect to this page, or have an `<iframe>` that references the page. You may also customize the `/opt/novell/nam/mag/webapps/nesp/jsp/logoutSuccess.jsp` file to provide login links or instructions to your user.

```
<tr>
  <td> <iframe src="https://idp.labs.com:8443/nidp/jsp/clearCookieAuth.jsp"
width="0" height="0"> </td>
</tr>
```

where `idp.labs.com` is the URL of the Identity Server.

## 4.3.5 Limitations

Following are the limitations with the Persistent Authentication Class:

- User is authenticated even if the password is changed.
- If the user is already logged in with **Remember Me** option enabled, you will be unable to stop the session until the cookie expires.

## 4.4 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**.

**Create Authentication Class**


Step 2 of 2: Specify properties.

**Servers**


New | Delete |  |  1 Item(s)


☐ Server


☐ 10.1.1.1

Port:  

Shared secret:

Reply time:   milliseconds

Resend time:   milliseconds

Failed server retry:   minutes

JSP:

User Lookup Attribute Name:

☐ Require password

<< Back Finish Cancel

- 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.
  - User Look Attribute Name:** Specify the LDAP attribute on which the user will be searched in the Radius server. CN is the default attribute.
  - 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 125](#).
- 10 Create a contract for the method:  
For instructions, see [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).  
If you want the user’s credentials 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 4.9, “Configuring Password Retrieval,” on page 166](#).
- 11 Update the Identity Server.

## 4.5 Configuring Client Integrity Check

You can configure a client integrity check policy to verify the recommended software (such as firewall and antivirus software) is installed on the client machine. You can configure different policies for Windows, Linux, and Macintosh machines and specify software that must be available in client machines to pass the client integrity check.

You need to create an identity provider authentication class that checks for the specified software on the client machine. You can configure policies to check processes, files, Windows registry, system services, and so on. This class can be executed with the first method of the contract. If the check fails, the user authentication fails.

Perform the following steps to configure the client integrity check:

- 1 Copy the following file from `/opt/novell/nam/sslvpn/webapps/sslvpn` to `/opt/novell/nam/idp/webapps/nidp/classUtils`:
  - For Linux:** LinCic from the linux folder.
  - For Macintosh:** MacCic from the MacOS and Maci386 folders.
  - For Windows:** wincic.msi from the windows folder.
- 2 Extract the binary file on the Identity Server machine and make it executable by using the `chmod +x`.

For example, run `chmod +x wincic.exe` for Windows.

- 3 Create CIC policies and assign it to a security level.

For more information about how to configure a CIC policy, see [“Configuring Policies to Check the Integrity of the Client Machine”](#) in the *NetIQ Access Manager 4.0 SSL VPN Server Guide*.

- 4 Copy the `cic_linux.txt`, `cic_windows.txt`, and `cic_mac.txt` files from `/etc/opt/novell/sslvpn` to the respective windows, linux and mac folders at `/opt/novell/nam/idp/webapps/nidp/classUtils`.
- 5 In the Administration Console, click **Identity Server > Edit > Local > Classes > New**.
- 6 Specify a name for the class and select `ClientIntegrityCheckClass` in **Java class**.
- 7 Click **Next**.
- 8 Click **New** and specify property name and property value.
- 9 Click **OK > Finish**.
- 10 Create a method for this class and deselect **Identifies User** check box and set all other fields to default settings and click **OK**. For instructions, see [Section 3.3, “Configuring Authentication Methods,”](#) on page 125.
- 11 Go to the **Contracts** tab and select CIC method from the **Available Methods** list and click **OK**. For instructions, see [Section 3.4, “Configuring Authentication Contracts,”](#) on page 128.

## 4.6 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.

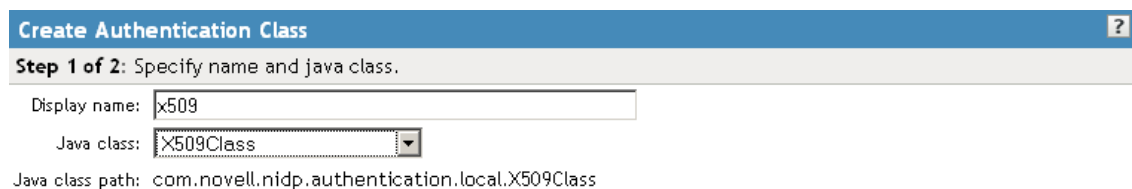
To configure X.509 authentication, you need to create an authentication class, then configure the validation and attribute mapping options.

- 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 about how to import trusted roots, see [“Importing Public Key Certificates \(Trusted Roots\)”](#) in the *NetIQ Access Manager 4.0 SP1 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**.



**Create Authentication Class** ?

**Step 1 of 2:** Specify name and java class.

Display name:

Java class:

Java class path: com.novell.nidp.authentication.local.X509Class

- 5 Specify a display name, then select **X509Class** from the drop-down menu.
- 6 Click **Next**.

Create Authentication Class

?

Step 2 of 2: Specify validations.

Validations

OCSP-CRL

CRL Validation

☐ Map X500 CRL to LDAP
 

LDAP URL:

OCSP Validation

☐ Sign OCSP request
 ☐ Use configured OCSP responder URL
 

URL:

☐ Disable Root CA revocation check

Trust Stores

2 Item(s)

Trust Store

[NIDP Trust Store](#)
[OCSP Trust Store](#)

Browser Restart

☒ Force browser restart on logout

<< Back

Finish

Cancel

## 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, select **Force browser restart on logout**.

**10** Click **Next**.

**11** Continue with [Section 4.6.1, “Configuring Attribute Mappings,” on page 159](#).

## 4.6.1 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.

## Create Authentication Class

### Step 3 of 3: Specify attribute mappings.

☐ Show certificate errors

☐ Auto Provision X509

Attributes:

Subject name



Available attributes:

Directory name  
Email  
Serial number and issuer name

#### Attribute Mappings

Directory name: sasAllowableSubjectNames

Email: mail

Serial number and issuer name:

Subject name: sasAllowableSubjectNames

**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 125](#).

5 Create a contract for the method:

For instructions, see [Section 3.4, "Configuring Authentication Contracts," on page 128](#).

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 4.9, "Configuring Password Retrieval," on page 166](#).

6 Update the Identity Server.

## 4.6.2 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 *NetIQ Access Manager 4.0 SP1 Administration Console Guide*.)
- 2 Create an X.509 authentication class. (See [Section 4.6, “Configuring Mutual SSL \(X.509\) Authentication,”](#) on page 157.)
- 3 Create an authentication method using this class. (See [Section 3.3, “Configuring Authentication Methods,”](#) on page 125.)
- 4 Create an authentication contract using the X.509 method. (See [Section 3.4, “Configuring Authentication Contracts,”](#) on page 128.)
- 5 Update the Identity Server cluster configuration. (See [Section 16.1.1, “Updating an Identity Server Configuration,”](#) on page 412.)
- 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 *NetIQ Access Manager 4.0 SP1 Access Gateway Guide*.
- 8 Update the Access Gateway.

## Customizing Certificate Errors

In case of certificate validation failure, the browser displays a standard `Page expired` error. If you want the Identity Server to display an Access Manager error instead of the usual error messages provided by the browser, edit the `/opt/novell/nam/idp/conf/server.xml` by using the following procedure:

- 1 Search for the `clientauth` attribute in the `server.xml` file.
- 2 Modify the value of the `clientauth` attribute from the default value of `false` to `want`.
- 3 Save the file and restart Identity Server by using the `rcnovell-idp restart` command.  
This setting ensures that the certificate is exchanged between the client and the server. This will result in a prompt being displayed on the browser during authentication.
- 4 Export the user and server certificate from the Administration Console by using the **Security > Certificates** option.  
To avoid the untrusted certificate messages in browsers, import the trusted root certificate of the CA into your browsers. For details, see “[Resolving Certificate Import Issues](#)” in the *NetIQ Access Manager 4.0 SP1 Administration Console Guide*

### 4.6.3 Configuring X.509 Authentication to Provide Access Manager Error Message

You can configure the X.509 class property and the Identity Server to avoid the browser provided message and display Access Manager error message. This occurs when the SSL mutual handshake fails because of non availability of client certificate.

---

**NOTE:** You can specify the Port and URL name as per your environment. The URL name and port number specified in the following procedure is an example.

---

**Prerequisite:** To configure you should have a parent domain, for example, `https://240onbox.provo.novell.com:8443/nidp/` and sub domain `https://onbox.provo.novell.com:8448/` available. You can also have a different port or IP combination.

Follow the procedure given below to configure X.509 based authentication for Access Manager specific error.

- 1 Go to identity provider and navigate to `/opt/novell/nam/idp/conf` directory.
- 2 Open `server.xml` file using the vi editor.
- 3 Search for `<Connector NIDP_Name="connector"` and create a copy of existing connector.
- 4 Go to the new connector you created and search for `clientAuth=false` string and change it to `clientAuth=want`. Change the port to a new port for example, 8448.
- 5 Save the `server.xml` file and exit.
- 6 Navigate to `/opt/novell/nids/lib/webapp/META-INF/` directory and open `context.xml` file.
- 7 Change Tomcat `context.xml` to set a same cookie for sub domains. Ensure that the path is set to `"/` as follows.

```
<?xml version="1.0" encoding="UTF-8"?>

    <Context sessionCookiePath="/"

        sessionCookieDomain=".provo.novell.com">                                <!-- Disable session
persistence across Tomcat restarts -->

        <Manager pathname="" saveOnRestart="false"/>

    </Context>
```

- 8 Change session proxying for setting this cookie.
  - 8a Navigate to `/opt/novell/nam/idp/webapps/nidp/WEB-INF/classes` directory
  - 8b Open `nidpconfig.properties` using vi editor and add the following lines.

```
CLUSTER_COOKIE_DOMAIN = .provo.novell.com
CLUSTER_COOKIE_PATH   = /
```

---

**NOTE:** Before you go to the next step, ensure that you have configured X.509 class, method, and contract. For more information about configuring, see [Section 4.6, "Configuring Mutual SSL \(X.509\) Authentication," on page 157](#).

---

- 9 For X509Class based redirection, in the X.509 method add the property for X.509 to be redirected to the new connector as follows.

Property Name: `CONNECTOR_HOST`

Property Value: `https://onbox.provo.novell.com:8448/`

- 10 Restart Tomcat using the following commands.

- ♦ Linux: `/etc/init.d/novell-idp restart`
- ♦ Windows: Enter the following commands:
  - ♦ `net stop Tomcat7`
  - ♦ `net start Tomcat7`

Verify the configuration as follows.

Access NIDP URL in browser that does not have client certificate. Access the X.509 authentication card and verify the behavior. It must redirect to connector port 8448 and redirect back to original page with an Access Manager error message or error code.

## 4.7 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 123](#).
  - 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 4.4, “Configuring for RADIUS Authentication,” on page 155](#).
- 8 (Conditional) If you selected the **Use X509** option, configure how the certificate is validated.

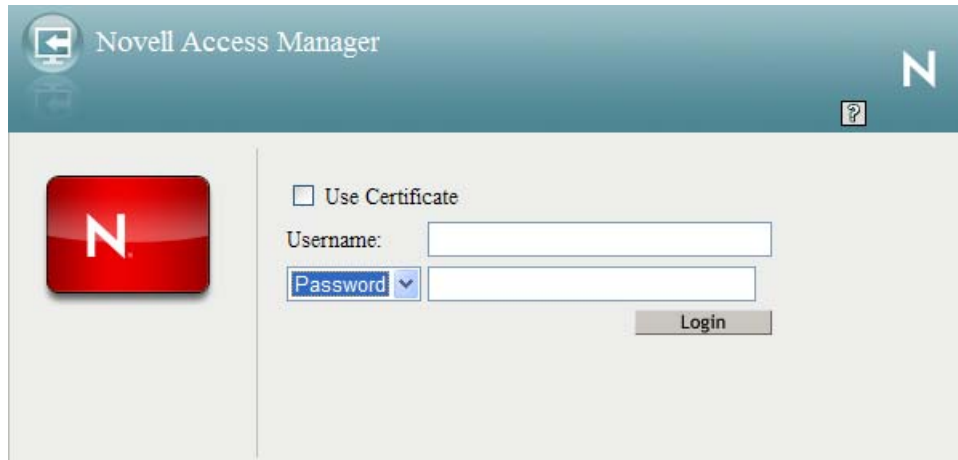
For information about the configuration options, see [Section 4.6, “Configuring Mutual SSL \(X.509\) Authentication,” on page 157](#).
- 9 Click **Next**.
- 10 (Conditional) If you selected the **Use X509** option, configure the attribute mappings.

For information about the configuration options, see [Section 4.6, “Configuring Mutual SSL \(X.509\) Authentication,” on page 157](#).

- 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 125](#) and [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).

If the contract allows the user to select from the three types of credentials, the login page looks similar to the following:

The image shows the Novell Access Manager login interface. At the top, there is a header bar with the Novell logo and the text "Novell Access Manager". Below the header, on the left, is a large red square button with a white "N" logo. To the right of this button, there is a login form. The form includes a checkbox labeled "Use Certificate". Below this, there are two input fields: "Username:" and "Password:". The "Password:" field has a small blue dropdown menu next to it. At the bottom right of the form is a "Login" button.

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.

## 4.8 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.

---

**NOTE:** Access Manager supports OpenID1.1.

---

- 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 125](#).

6 Create a contract for the method:

For instructions, see [Section 3.4, "Configuring Authentication Contracts," on page 128](#).

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 4.9, "Configuring Password Retrieval," on page 166](#).

7 Update the Identity Server.

## 4.9 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:

**passwordfetchclass**

---

General Properties

---

**General settings**

☐ Ignore password retrieval failure

☒ Retain previous principal

Password to be retrieved

☒ Universal Password

☐ Simple Password

**Userstore lookup settings**

User lookup method

☒ Based on the CN of the user object

☐ Based on the attribute value of the user object

Attribute details

Attribute name of the DN

☐ Auto Provision

---

OK Cancel Apply

---

**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.

**Retain Previous Principal:** Select this option to retain the principal obtained from the previous authentication method. If you do not select this option, then the principal will be used from the method associated with this class.

**Password to be retrieved:** If your users have been configured to use a universal password, select **Universal Password**. Otherwise, select **Simple Password**.



---

**NOTE**

- ♦ Set the Universal Password Retrieval options in the configuration of the Universal Password policy, so that the policy allows the password to be retrieved from the User Store.
  - ♦ User must reset the password after configuring the password policy for universal password.
- 

For more information about Unable to retrieve Universal Password from eDirectory using PasswordFetchClass issue, see [TID 7007114](#).

The password retrieval of the user store lookup settings are mapped based on the CN attribute value of the user object, attribute details that is based on the distinguished name to LDAP attribute. These options are added so that the password can be fetched from Active Directory. If you are not using Active Directory but eDirectory, then you use the default CN attribute of the eDirectory to map with the Active Directory user store.

**4** Configure the following UserStore Lookup Settings:

**Based on the CN of the user object:** The CN users are mapped between two different user stores. CN is mapped with for retrieving the password from user store. For Example - Active Directory CN is mapped with eDirectory CN for retrieving the password from eDirectory user store.

**Based on the Attribute value of the user object:** The user names are detected and handled in LDAP attribute or DN users of the Active Directory are mapped with LDAP attribute of the eDirectory. If you select this option, then specify the attribute value in attribute details of the **Attribute name of DN** and enable the **Auto Provision** check box if required.

**Attribute Name of the DN:** Specify the attribute name of the DN.

This attribute must contain the CN of user whose password you want to obtain. For example, if you are trying to obtain a password from eDirectory for a user with cn=a,dc=b, then you need to specify name of the attribute, which value is cn=a,dc=b. The passwordfetchclass tries fetching the password from the current user store based on the value of the LDAP attribute specified, which are mapped to user's DN of the Active Directory.

**Auto Provision:** If you enable this check box, then the passwordfetchclass tries fetching the password from LDAP attribute specified above 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 value gets populated with the DN user of the Active Directory. When the user is logged next time the same value is used.

**5** Click **OK**.

**6** Create a method for this class.

For instructions, see [Section 3.3, “Configuring Authentication Methods,” on page 125](#).

**7** 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 155](#).
- ♦ X.509. See [“Configuring Mutual SSL \(X.509\) Authentication” on page 157](#).
- ♦ OpenID. See [“Configuring for OpenID Authentication” on page 165](#).
- ♦ Smart Card. See [“Configuring Access Manager for NESCM” on page 169](#).
- ♦ Kerberos. See [“Configuring for Kerberos Authentication” on page 177](#).
- ♦ Google Authenticator. See [“Configuring for Two-Factor Authentication Using Time-Based One-Time Password \(TOTP\)” on page 150](#)

**8** Click **Apply** and update the Identity Server.



## 4.10 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 4.10.1, “Prerequisites,” on page 169](#)
- ♦ [Section 4.10.2, “Creating a User Store,” on page 169](#)
- ♦ [Section 4.10.3, “Creating a Contract for the Smart Card,” on page 171](#)
- ♦ [Section 4.10.4, “Assigning the NESCM Contract to a Protected Resource,” on page 175](#)
- ♦ [Section 4.10.5, “Verifying the User’s Experience,” on page 175](#)
- ♦ [Section 4.10.6, “Troubleshooting,” on page 176](#)

### 4.10.1 Prerequisites

- ❑ Ensure that 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](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](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](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](http://www.novell.com/documentation/iasclient30x/nescm_install/data/bookinfo.html)).
  - ♦ Provision your smart card according to your company policy.
- ❑ Ensure that 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 *NetIQ Access Manager 4.0 SP1 Installation Guide* and the *NetIQ Access Manager 4.0 SP1 Setup Guide*.

### 4.10.2 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 4.10.3, “Creating a Contract for the Smart Card,” on page 171](#).
- ♦ 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**.

### Create User Store

Specify name, administrator, password and search contexts

Name: \*

Admin name: \*

(Ex: cn=admin,o=novell)

Admin password: \*

Confirm password: \*

Directory type: eDirectory

☐ Install NMAС SAML method

☐ Enable Secret Store lock checking

### LDAP timeout settings

LDAP Operation:  seconds

Idle Connection:  seconds

### Server replicas

[New](#) | [Delete](#) | [Validate](#)

<input type="checkbox"/> Name	IP Address	Port	Use SSL	Max. Connections	Validation Status
No items					

### Search Contexts

[New](#) | [Delete](#) | [Up](#) | [Down](#)

<input type="checkbox"/> Context	Scope
No items	

- 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.

---

**NOTE:** If the admin account's password needs to be changed in the LDAP directory due to some issue, then change the admin password in the Create User Store page accordingly and apply the change. Else, this admin account of the user store will get locked.

---

**Directory Type:** Select eDirectory.

- 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.

- 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.
- Click **Auto import trusted root**.
- Click **OK** to confirm the import.
- Select the **Root CA Certificate** to trust any certificate signed by that certificate authority.
- 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 4.10.3, “Creating a Contract for the Smart Card,” on page 171](#).

### 4.10.3 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 171](#)
- ♦ [“Creating a Method to Use the NMAS Class” on page 172](#)
- ♦ [“Creating an Authentication Contract to Use the Method” on page 173](#)

#### 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**.

The screenshot shows a dialog box titled "Create Authentication Class" with a subtitle "Step 1 of 2: Specify name and java class." It contains three fields: "Display name:" with the text "Class-NMAS-NESCM", "Java class:" with a dropdown menu showing "NMASAuthClass", and "Java class path:" with the text "com.novell.security.nmas.nidp.NMASAuthClass".

- 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**.

**Create Authentication Class**

**Step 2 of 2:** Specify properties.

[New](#) | [Delete](#)

<input type="checkbox"/> Name	Value
<i>No items</i>	

**Add property** ✕

Property Name:

Property Value:

- 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 172](#).

## 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**.

**Create Authentication Method** ?

**Configuration**

Display name:

Class:

☒ Identifies User

User stores:

Available user stores:

**Properties**

New | Delete 0 Item(s)

<input type="checkbox"/> Name	Value
No items	

<< Back Finish Cancel

- 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 171](#).
- 4 In the **Available user stores list**, select the user store created in [Section 4.10.2, “Creating a User Store,” on page 169](#), 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 173](#).

## 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 172, 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 4.9, “Configuring Password Retrieval,”](#) on page 166.

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 4.10.4, “Assigning the NESCM Contract to a Protected Resource,”](#) on page 175

## 4.10.4 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 *NetIQ Access Manager 4.0 SP1 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.

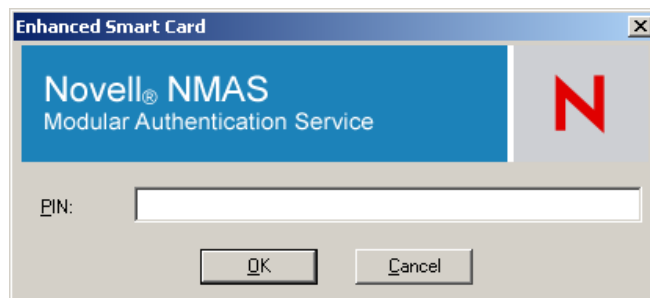
- 2 Click the **Protected Resource** link for the proxy service where you want to assign the NESCM contract.
- 3 To enable the NESCM contract on an existing protected resource, click the **Authentication Procedure** link for that resource, then select the NESCM contract created in “[Creating an Authentication Contract to Use the Method](#)” on page 173.

If the contract is not listed, ensure that 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, ensure 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 4.10.5, “Verifying the User’s Experience,”](#) on page 175.

## 4.10.5 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.

## 4.10.6 Troubleshooting

Error	Resolution
Authentication fails without prompting the user for the token	Verify that you have configured the class and method correctly. See <a href="#">“Creating an NMAS Class for NESCM” on page 171</a> and <a href="#">“Creating a Method to Use the NMAS Class” on page 172</a> .
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.



# 5 Configuring for Kerberos Authentication

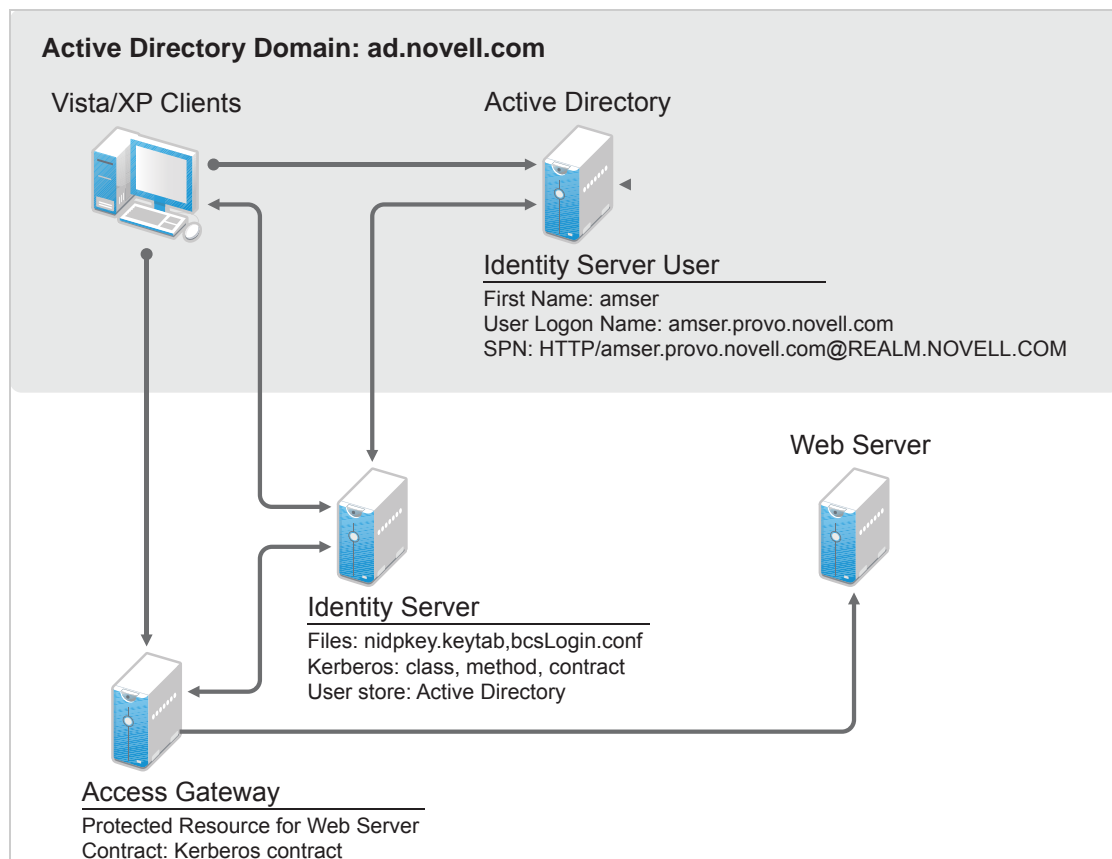
Kerberos is an authentication method that allows users to log in to an Active Directory domain. This authentication method provides them with a token, which an Identity Server can be configured to use as a contract. This provides single sign-on for the user between Active Directory and the Identity Server.

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** Example Kerberos Configuration



Kerberos requires the following configuration tasks:

- ♦ [Section 5.1, “Prerequisites,” on page 178](#)
- ♦ [Section 5.2, “Configuring Active Directory,” on page 178](#)
- ♦ [Section 5.3, “Configuring the Identity Server,” on page 181](#)
- ♦ [Section 5.4, “Configuring the Clients,” on page 188](#)
- ♦ [Section 5.5, “Configuring the Access Gateway for Kerberos Authentication,” on page 190](#)

## 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 about how to enable this feature, see [“Authentication Uses NTLM instead of Kerberos” \(http://technet.microsoft.com/en-us/library/cc779070.aspx\)](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 180](#).

For more information about these issues, see [TID 7006036 \(http://www.novell.com/support/viewContent.do?externalId=7006036&slicId=1\)](http://www.novell.com/support/viewContent.do?externalId=7006036&slicId=1).

- ♦ Active Directory must be configured to contain entries for both users and their machines. Active Directory must be running on Windows Server 2003 Enterprise SP2, Windows Server 2008, Windows Server 2008 R2, or Windows Server 2012.
- ♦ 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.

## 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:

- ♦ [Section 5.2.1, “Installing the spn and the ktpass Utilities for Windows Server 2003,” on page 179](#)
- ♦ [Section 5.2.2, “Creating and Configuring the User Account for the Identity Server,” on page 179](#)
- ♦ [Section 5.2.3, “Configuring the Keytab File,” on page 180](#)
- ♦ [Section 5.2.4, “Adding the Identity Server to the Forward Lookup Zone,” on page 180](#)

## 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 for this user, open a command window and enter the following commands:

```
setspn -A HTTP/<userLogonName> <userName>
```

For this configuration example, you would enter the following command:

```
setspn -A HTTP/amser.provo.novell.com@AD.NOVELL.COM amser
```

This adds the servicePrincipalName attribute to the user specified with the value specified in the `-A` parameter.

---

**NOTE:** For Domain Services for Windows, set HOST spn also by using this command: `setspn -A HOST/<userLogonName> <userName>`

---

- 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 <code>.keytab</code> as the extension. For example: <code>nidpkey.keytab</code>
/princ	<servicePrincipalName> @<KERBEROS_REALM>	Specify the service principal name for the Identity Server, then <code>@</code> , 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 file to the default location on the Identity Server:

**Linux:** `/opt/novell/java/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. See the [NetIQ Access Manager 4.0 SP1 Installation Guide](#) and the [NetIQ Access Manager 4.0 SP1 Setup Guide](#).

- ♦ [Section 5.3.1, “Enabling Logging for Kerberos Transactions,” on page 181](#)
- ♦ [Section 5.3.2, “Configuring the Identity Server for Active Directory,” on page 181](#)
- ♦ [Section 5.3.3, “Creating the Authentication Class, Method, and Contract,” on page 182](#)
- ♦ [Section 5.3.4, “Creating the bcsLogin Configuration File,” on page 185](#)
- ♦ [Section 5.3.5, “Verifying the Kerberos Configuration,” on page 186](#)
- ♦ [Section 5.3.6, “\(Optional\) Using the Name/Password Form Authentication,” on page 186](#)
- ♦ [Section 5.3.7, “\(Optional\) Configuring the Fall Back Authentication Class,” on page 187](#)

### 5.3.1 Enabling Logging for Kerberos Transactions

Enabling logging 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 182](#).

### 5.3.3 Creating the Authentication Class, Method, and Contract

- 1 In the Local page, click **Classes > New**.

**Create Authentication Class**

**Step 1 of 2:** Specify name and java class.

Display name:

Java class:

Java class path:

- 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**.

Create Authentication Class

Step 2 of 2: Specify properties.

Service Principal Name (SPN): HTTP/amser.provo.novell.com

Kerberos Realm: AD.NOVELL.COM

JAAAS config file for Kerberos: /opt/novell/java/jre/lib/security/bcsLogin.conf

Kerberos KDC: 10.10.16.79

User Attribute: userprincipalname

UPN Suffixes

New | Delete

0 Item(s)

☐ Suffix

No items

#### 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.

**JAAAS 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 180) 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 185.

**Kerberos KDC:** Specify the IP address of the Key Distribution Center. If multiple KDCs are present for fail-over support, then specify the IP addresses separated by colon (:). Maximum of 4 IP addresses can be configured.

If a L4 switch is configured for load balancing between the KDCs, then enter the virtual IP address of the L4 switch in this field.

**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.

- (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**.

- Click **Finish**.

---

**IMPORTANT:** You should create only one Kerberos class. This is caused by a limitation in the underlying Sun JGSS.

---

- On the Local page, click **Methods > New**.

- 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**.

**Create Authentication Contract**

Configuration

Display name:

URI:

Password expiration servlet:

Authentication Level:

☐ Satisfiable by a contract of equal or higher level

☐ Satisfiable by External Provider

If you add more than one X509 method, only the first one will be used and it will automatically be moved to the top of the list.

Methods:

Available methods:

- Name/Password - Basic
- Name/Password - Form
- Secure Name/Password - Basic
- Secure Name/Password - Form

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 identify 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 185](#).

## 5.3.4 Creating the bcsLogin Configuration File

If you are upgrading from 3.1 SP2 to 3.1 SP3, the syntax of the `bcsLogin.conf` file has changed.

To create the file:

- 1 Open a text editor. A sample `bcsLogin.conf` file called `bcsLogin.conf.template` is included that can be edited. Open this file.
- 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";
};
```

The Identity Server will check the Kerberos server for each user transaction. When you set the `isInitiator` value to `false` (`isInitiator="false"`) in the `bcsLogin.conf` file after the `keyTab="/opt/novell/java/jre/lib/security/nidpkey.keytab"` line, the Identity Server does not communicate to the Kerberos server.

Path of `bcsLogin.conf` on SLES and Red Hat is `/opt/novell/java/jre/lib/security/`.

---

**NOTE:** Before setting the value to `"false"`, it is recommended that you access the protected site via `https` and keytab file is secure.

---

For Windows, the path needs to contain double slashes: `C:\\Program Files\\Novell\\jre\\lib\\security`

**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 182.
- 5 Ensure that the file permissions are set correctly. They should be set to 644.
- 6 Restart Identity Server:
  - ♦ **Linux Identity Server:** Enter the following command:
 

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

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

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 `catalina.out` (Linux) or the `stdout.log` (Windows) 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 181.
- 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.

### 5.3.6 (Optional) Using the Name/Password Form Authentication

You can configure the IP address or the range of IP addresses of the clients for which the kerberos authentication should be skipped or performed using the `kerberos.exclude` or `kerberos.include` keywords respectively.

---

**NOTE:** You can specify only `kerberos.exclude` or `kerberos.include` argument in the `kerb.properties` file not both.

---

To configure this option, add the following entry in the `kerb.properties` file:

- ♦ `kerberos.exclude`=IP Address/Range separated by comma.
- ♦ `kerberos.include`=IP Address/Range separated by comma.

For example:

```
kerberos.exclude=1.1.1.1-9.255.255.255,10.50.1.1 - 10.50.1.50,11.1.1.1-255.255.255.255
```

or

```
kerberos.include=10.1.1.1-10.49.255.255,10.50.1.51-10.255.255.255
```

For the clients coming from the IP addresses specified in `kerberos.exclude`, Kerberos authentication will be skipped and will fall back to the custom authentication class. See [Section 5.3.7, “\(Optional\) Configuring the Fall Back Authentication Class,” on page 187](#)

For the clients coming from the IP addresses that are not specified in `kerberos.include`, kerberos authentication will be skipped and will fall back to the custom authentication class. See [Section 5.3.7, “\(Optional\) Configuring the Fall Back Authentication Class,” on page 187](#)

The `kerb.properties` file is available at the following locations:

**Linux:** `/var/opt/novell/nids/lib/webapp/WEB-INF/classes/kerb.properties`

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

## 5.3.7 (Optional) Configuring the Fall Back Authentication Class

You can configure an optional authentication class that has to be executed when either kerberos authentication fails or when kerberos authentication has to be skipped.

For information about how to skip the kerberos authentication for certain IP addresses, see [“\(Optional\) Using the Name/Password Form Authentication” on page 186](#)

To configure the fall back authentication class:

- 1 Go to the **Identity Server Cluster > Edit > Local > Methods > (Kerberos Method) > Properties** tab.
- 2 Add a new property /value pair with name as `FALLBACK_AUTHCLASS` and set the property value to be the qualified class name such as `com.novell.nidp.authentication.local.PasswordClass`.

The class name value should be same as the value configured in the Java class path of the class at **IDP Cluster > Edit> Local > Classes> (Authentication class)**.

---

**NOTE:** If your authentication class requires a custom JSP file for seeking credentials, add the property `JSP` and specify the name of the jsp file. When the JSP property is not specified, Identity Server will use the default `login.jsp` for seeking the credentials.

---

If you want to fall back to basic authentication, configure any one of the following properties:

Property Name: `FALLBACK_AUTHCLASS`

Property Value: Basic or com.novell.nidp.authentication.local.BasicClass

---

**NOTE:** The property name is case-sensitive.

---

For example, if you want to fall back to Radius, configure the following properties for the kerberos method:

FALLBACK\_AUTHCLASS=com.novell.nidp.authentication.local.RadiusClass

JSP=radiuslogin

Server=<<radius IPs with comma separate>>

SharedSecret=<<secret string>>

Port=<<port>>

ReplyTime=7000 (in milli seconds, this is optional)

ResendTime=2000 (in milli seconds, this is optional)

Retry=5 (this is optional)

Password=false

---

**NOTE:** The property name is case-sensitive.

---

Also, you can configure fall back to other mechanism based on the incoming header. In the kerberos Method, add the name/value in the property field with name as NO\_NEGO\_HEADER\_NAME and in the value field you can provide the header, which needs to be ignored for the kerberos authentication.

For Example, in the kerberos method properties, if you configure the name as NO\_NEGO\_HEADER\_NAME with value X-NovINet. Then if the client comes with header X-NovINet, the kerberos class will not be executed and it will fall back to the name password form by default or to the configured fall back mechanism.

For more information about using this feature, see [Cool Solution \(https://www.netiq.com/communities/cool-solutions/hold-howto-single-sign-with-netidentity-novell-access-manager/\)](https://www.netiq.com/communities/cool-solutions/hold-howto-single-sign-with-netidentity-novell-access-manager/)

## 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 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 186](#).
  - ♦ 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 *NetIQ Access Manager 4.0 SP1 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 4.9, “Configuring Password Retrieval,”](#) on page 166.

---

# 6 Defining Shared Settings

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.
- ♦ **Metadata Repositories:** Import metadata of trusted providers.

These features are configurable from the **Shared Settings** tab on the Identity Servers page.

This section describes the following tasks:

- ♦ [Section 6.1, “Configuring Attribute Sets,” on page 191](#)
- ♦ [Section 6.2, “Editing Attribute Sets,” on page 194](#)
- ♦ [Section 6.3, “Configuring User Matching Expressions,” on page 194](#)
- ♦ [Section 6.4, “Adding Custom Attributes,” on page 196](#)
- ♦ [Section 6.5, “Adding Authentication Card Images,” on page 199](#)
- ♦ [Section 6.6, “Creating an Image Set,” on page 199](#)
- ♦ [Section 6.7, “Metadata Repositories,” on page 200](#)

## 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**.

**Create Attribute Set** ?

**Step 1 of 2:** Name attribute set

Set Name

Select set to use as template

- 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**.

**Create Attribute Set**

**Step 2 of 2:** Define attributes

[New](#) | [Delete](#)

☐ Local Attribute maps to Remote Attribute

*No items*

**Add Attribute Mapping** ✕

☒ Local attribute:  ▼

☐ Constant:

Remote attribute:  (optional)

Remote namespace: ☒ none ▼

☐

Remote format:  ▼

OK Cancel

- 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 196.



- ♦ **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 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:

Identity Servers ►

Create Attribute Set <span>?</span>		
Step 2 of 2: Define attributes		
New   Delete		1 Item(s)
<input type="checkbox"/> Local Attribute	maps to	Remote Attribute
<input type="checkbox"/> Common First Name [Personal Profile]	<-->	firstname

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 7.8, “Selecting Attributes for a Trusted Provider,” on page 224](#).

## 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 191.](#))

- 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.

**First Name** ?

**General** **Mapping** Usage

[New](#) | [Delete](#) 1 Item(s)

<input type="checkbox"/> Local Attribute	maps to	Remote Attribute
<input type="checkbox"/> Every Day Name [Personal Profile]	<-->	First Name

- 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.)

The screenshot shows the 'User Matching Expression' configuration window. It has a title bar 'User Matching Expression' and a toolbar with 'New Logic Group' and 'Delete'. Below the toolbar, there's a 'Groups' section with a plus icon and a 'Type' dropdown. The 'Logic Group 1' is expanded, showing a list of attributes: Common Personal Title, Common First Name, Common Last Name, Common Middle Name, Legal Name, Legal Middle Name, and Legal Fiscal Identification Type. An 'Add Attributes' dialog box is open, showing a list of attributes to add: Informal Name, Every Day Name, Common Personal Title, Common First Name, Common Last Name, Common Middle Name, Legal Name, and Legal Personal Title. The dialog has 'OK' and 'Cancel' buttons. At the bottom of the main window, there are 'OK', 'Cancel', and 'Apply' buttons.

- 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.

- ♦ [Section 6.4.1, “Creating Shared Secret Names,” on page 196](#)
- ♦ [Section 6.4.2, “Creating LDAP Attribute Names,” on page 197](#)

### 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 an External Attributes policy, **Secret Name** should match the policy name and **Secret Entry Name** should match the attribute name configured while creating the policy.

For example, if the policy name is fetchattr and attribute name configured in the policy is address, then **Secret Name** should be fetchattr and **Secret Entry Name** should be address.

For more information about how to use shared secrets with policies, see “[Creating and Managing Shared Secrets](#)” in the *NetIQ Access Manager 4.0 SP1 Policy Guide*.

The Identity Server needs to be configured to use shared secrets. For information about this process, see [Section 3.1.4, “Configuring a User Store for Secrets,” on page 111](#).

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**.

#### Identity Servers

**Servers** **Shared Settings**

Attribute Sets | User Matching Expressions | **Custom Attributes** | Authentication Card Images

Add custom shared secret names or LDAP attribute names that you want to be selectable in policy select lists.

**Shared Secret Names**

[New](#) | [Delete](#)

<input type="checkbox"/> Name	Entries
<input type="checkbox"/> <a href="#">login</a>	name, password, employee ID

**LDAP Attribute Names**

[New](#) | [Delete](#) | [Set Encode](#) | [Clear Encode](#)

<input type="checkbox"/> Name	64-bit Encode Attribute Data
<input type="checkbox"/> audio	
<input type="checkbox"/> businessCategory	
<input type="checkbox"/> carLicense	
<input type="checkbox"/> cn	
<input type="checkbox"/> departmentNumber	
<input type="checkbox"/> description	
<input type="checkbox"/> displayName	
<input type="checkbox"/> employeeNumber	

**Shared Secret Names**

Enter a new Shared Secret Name.

Secret Name:

Secret Entry Name:

OK

- 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 ensure that 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 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 199](#).

## 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**.

## 6.7 Metadata Repositories

Large scale federations have more than 100+ identity and or service providers and it is a tedious task to establish bi-lateral relationships with Access Manager. You as an identity provider can now configure several identity and or service providers using a multi-entity metadata file available in a central repository. The identity and/or service providers become partners of a community which maintains a single metadata file containing metadata of all the approved partners. The identity and or service providers submit their metadata which includes specifications of services offered (SAML 1.1 and SAML 2.0) and any other information. This feature is available only for SAML 1.1 and SAML 2.0.

For example, XYZ is an e-book store and several e-book stores, which are either identity or service providers are partner with it. XYZ maintains a single metadata file containing metadata of all the other stores. ABC an e-book identity provider wants to establish a federation with many other e-book stores. Hence, ABC partners with XYZ by sharing its metadata and XYZ in turn shares the metadata XML file. ABC imports the XML file available publicly on the internet (for example, <http://xyz.commonfederation.org/xyz-metadata.xml>) and establishes trusts with others in the federation which includes XYZ's trusted provider sites.

### 6.7.1 Creating Metadata Repositories

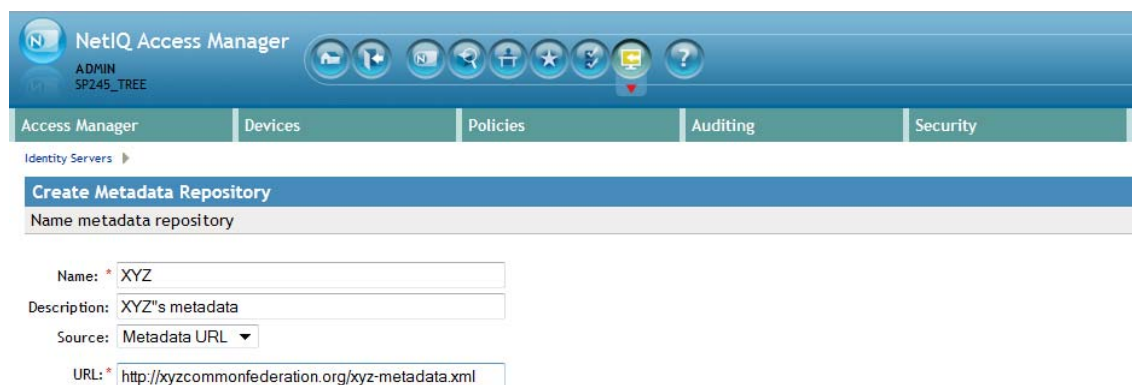
- 1 In the Administration Console, click **Devices > Identity Servers > Shared Settings > Metadata Repositories**.
- 2 Click **New** and fill in the following fields:

**Name:** Enter the name of the metadata repository.

**Description:** Enter the description of the metadata repository.

**Source:** From the drop-down menu, select the source from which you want to import the metadata file.

- ♦ To specify the URL location of the XML file in the **URL** field, select **Metadata URL**.
- ♦ To specify the path of the XML file in the **File** field, select **Metadata File**.



The screenshot shows the NetIQ Access Manager Administration Console interface. The top navigation bar includes 'Access Manager', 'Devices', 'Policies', 'Auditing', and 'Security'. Below this, the 'Identity Servers' section is active. The 'Create Metadata Repository' form is displayed, with the following fields:

- Name:** \* XYZ
- Description:** XYZ's metadata
- Source:** Metadata URL (selected from a dropdown menu)
- URL:** \* <http://xyzcommonfederation.org/xyz-metadata.xml>

- 3 Click **Finish**.

The details of the metadata such as the number of identity servers and service providers present in the metadata, and expiry date of the metadata are displayed.



You can select the metadata repository and click **Delete** to delete the repository. If the metadata file is in use, you cannot delete it. Delete the trusted provider first and then delete the metadata file.

- 4 Select **All** to see a list of entities. If the entity is supporting it the respective protocol will be checked.

Once the metadata repositories are imported, the entities available in the metadata repository can be assigned as trusted provider to any of the Identity Provider clusters. To create the trusted providers, see [Section 7.3, “Managing Trusted Providers,” on page 209](#).

## 6.7.2 Reimporting Metadata Repositories

You can reimport the metadata repository to get the updated XML.

- 1 In the Administration Console, click **Devices > Identity Servers > Shared Settings > Metadata Repositories**.
- 2 Click on the metadata repository you created and click **Reimport**.
- 3 Specify the URL location of the XML file in the **URL** field and click **Next**.
- 4 The screen displays the following:

**New Entities added to the repositories:** If the entities are updated or deleted and are assigned as Trusted

Providers to an Identity Server cluster then the Identity Server cluster name is displayed in brackets next to the entity ID.

**Entities Deleted from the repositories:** If the entity is updated and is assigned as a trusted provider to an Identity Server cluster, that trusted provider will be updated. You must update the Identity Server cluster for the changes to take effect.

**Entities Updated in the repositories:** If an entity is deleted and was assigned as trusted provider to an Identity Server cluster, then the link between the trusted provider and the metadata repository entity is deleted.

---

**NOTE:** The corresponding trusted provider is not deleted and you will have to manually delete the trusted provider.

---

- 5 Click **Finish** to apply the changes.



---

# 7 Configuring SAML and Liberty Trusted Providers

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).

- ♦ [Section 7.1, “Understanding the Trust Model,” on page 204](#)
- ♦ [Section 7.2, “Configuring General Provider Options,” on page 206](#)
- ♦ [Section 7.3, “Managing Trusted Providers,” on page 209](#)
- ♦ [Section 7.4, “Modifying a Trusted Provider,” on page 216](#)
- ♦ [Section 7.5, “Executing Authorization Based Roles Policy During SAML 2.0 Service Provider Initiated Request,” on page 217](#)
- ♦ [Section 7.6, “Contracts Assigned to SAML 2.0 Service Provider,” on page 218](#)
- ♦ [Section 7.7, “Configuring Communication Security,” on page 220](#)
- ♦ [Section 7.8, “Selecting Attributes for a Trusted Provider,” on page 224](#)
- ♦ [Section 7.9, “Managing Metadata,” on page 227](#)
- ♦ [Section 7.10, “Configuring an Authentication Request for an Identity Provider,” on page 232](#)
- ♦ [Section 7.11, “Configuring an Authentication Response for a Service Provider,” on page 237](#)
- ♦ [Section 7.12, “Routing to an External Identity Provider Automatically,” on page 242](#)
- ♦ [Section 7.13, “Defining Options for Liberty or SAML 2.0,” on page 242](#)
- ♦ [Section 7.14, “Defining Options for SAML 1.1 Service Provider,” on page 245](#)
- ♦ [Section 7.15, “Defining Session Synchronization for the A-Select SAML 2.0 Identity Provider,” on page 245](#)
- ♦ [Section 7.16, “Configuring the Liberty or SAML 2.0 Session Timeout,” on page 246](#)
- ♦ [Section 7.17, “Managing the Authentication Card of an Identity Provider,” on page 247](#)
- ♦ [Section 7.18, “Using the Intersite Transfer Service,” on page 248](#)
- ♦ [Section 7.19, “Enabling or Disabling SAML Tags,” on page 257](#)
- ♦ [Section 7.20, “Sample Configurations,” on page 259](#)
- ♦ [Section 7.21, “Configuring Multiple SAML 2.0 Service Providers on the Same Host for a Single SAML Identity Provider,” on page 264](#)

## About SAML and Liberty

For information about how Access Manager uses SAML, see [Appendix B, “Understanding How Access Manager Uses SAML,” on page 481](#).

For conceptual information about Liberty, see [Appendix A, “About Liberty,” on page 479](#).

## 7.1 Understanding the Trust Model

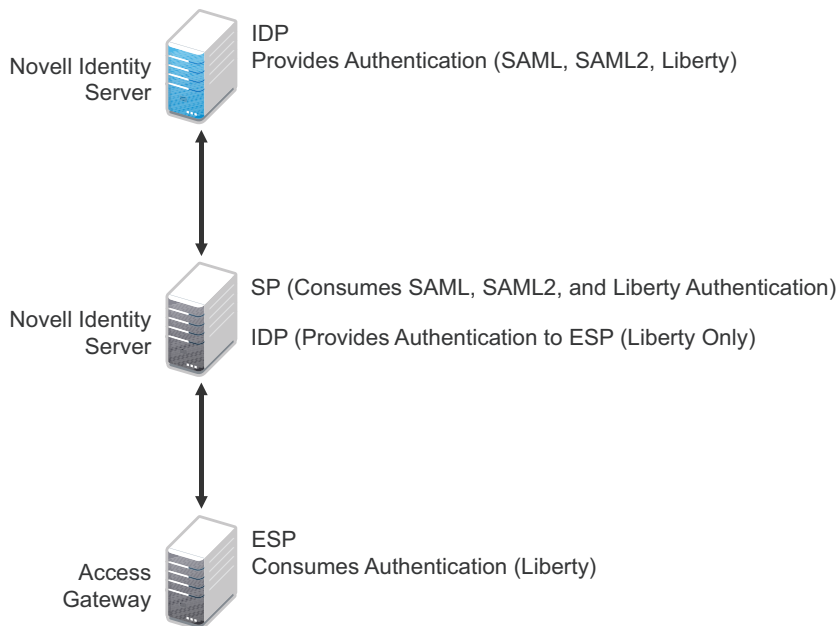
Setting up trust involves system administrators agreeing on how to establish a secure method for providing and consuming authentication assertions between their Identity Servers. An Identity Server is always installed as an identity provider, which is used to provide authentication to trusted service providers and ESPs. It can also be configured to be a service provider and trust the authentication of an identity provider.

- ♦ [Section 7.1.1, “Identity Providers and Consumers,” on page 204](#)
- ♦ [Section 7.1.2, “Embedded Service Providers,” on page 205](#)
- ♦ [Section 7.1.3, “Configuration Overview,” on page 205](#)

### 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

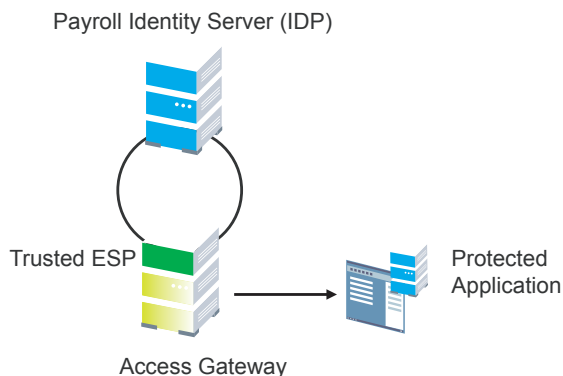


As an administrator, you determine whether your server is to be used as the identity provider or service provider in the trust relationship. You and the trusted partner agree to exchange identity provider metadata, and then you create references to the trusted partner's identity provider or service provider 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 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 [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 18](#). (You should already be familiar with the [NetIQ Access Manager 4.0 SP1 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 Service Provider for SAML 2.0,” on page 210](#).

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 13, “Configuring User Identification Methods for Federation,” on page 369](#).

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 128](#).
  - ♦ Enabling the profiles that you are using. See [Section 15.2, “Managing Web Services and Profiles,” on page 386](#).
  - ♦ Enabling the **Always Allow Interaction** option on the Web Service Consumer page. See [Section 15.5, “Configuring the Web Service Consumer,” on page 396](#).
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 251](#).

---

**NOTE:** For a tutorial that explains all the steps for setting up federation between two NetIQ Identity Servers, see [“Setting Up Federation”](#) in the *NetIQ Access Manager 4.0 SP1 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:

- ♦ [Section 7.2.1, “Configuring the General Identity Provider Options,” on page 206](#)
- ♦ [Section 7.2.2, “Configuring the General Identity Consumer Options,” on page 207](#)
- ♦ [Section 7.2.3, “Configuring the Introductions Class,” on page 208](#)
- ♦ [Section 7.2.4, “Configuring the Trust Levels Class,” on page 209](#)

### 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 1.4.4, “Managing the Keys, Certificates, and Trust Stores,” on page 31](#).

- 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 208](#).

---

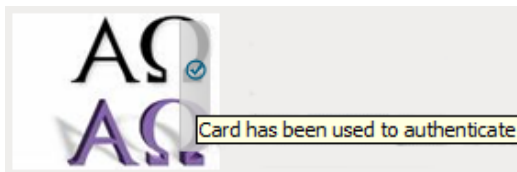
**SSL Certificate:** Displays the Keystore page that you use to locate and replace the test-consumer SSL certificate for this configuration.

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 1.4.4, “Managing the Keys, Certificates, and Trust Stores,” on page 31](#).

- 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 a 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 207](#).

## 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 137](#) and [Section 3.5.1, “Specifying Authentication Types,” on page 138](#).

- 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 NetIQ 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 Service Provider for SAML 2.0,” on page 210](#)
- ♦ [Section 7.3.2, “Creating a Trusted Service Provider for SAML 1.1 and Liberty,” on page 212](#)
- ♦ [Section 7.3.3, “Creating a Trusted Identity Provider,” on page 214](#)

**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 Service Provider for 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 1.4.4, "Managing the Keys, Certificates, and Trust Stores," on page 31](#).
- Shared the trusted root of the SSL certificate of your Identity Server with the other provider so that the administrator can import 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 **SAML 2.0**.

General Local Liberty SAML 1.1 SAML 2.0 WS Federation Brokering WS-Trust				
Trusted Providers   Profiles				
New   Delete   Enable   Disable				
<input type="checkbox"/>	Name	Enabled	Metadata Expiration Date	Metadata Repository
Identity Providers				
<input type="checkbox"/>	<a href="https://gmc.89-205.labs.blr.novell.com/nidp/saml2/metadata">https://gmc.89-205.labs.blr.novell.com/nidp/saml2/metadata</a>	✓	Not specified	289CMD

- Click **New**, then click **Service Provider**.

### Create Trusted Service Provider

#### Step 1 of 2: Specify name and metadata

Provider Type:

General

Source:

Metadata URL

Name:

\*

URL:

\*

**NOTE:** By default, the **Provider Type General** is selected. You can configure an Identity Server to trust a service provider to establish federation with external service providers. For more information on pre-configured metadata for Google Applications, Office 365, and Salesforce.com, see [Section 7.20, “Sample Configurations,” on page 259](#).

- 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`
- ♦ **OIOSAML:** `http://10.1.1.1/nidp/saml2/metadata_oiosaml`
- ♦ **OIOSAML:** `https://10.1.1.1/nidp/saml2/metadata_oiosaml`

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 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 page to enter the required details. See [“Editing a SAML 2.0 Service Provider’s Metadata” on page 231](#).

**Metadata Repositories:** Allows you to configure several identity and/or service providers using a multi-entity metadata file available in a central repository. For more information about creating Identity and/or Service Providers see, [Section 7.3.4, “Creating Identity Providers and Service Providers,” on page 215](#).

- 4 In the **Name** option, specify a name by which you want to refer to the provider.
- 5 Click **Next**.
- 6 Review the metadata certificates and click **Finish**. The system displays the trusted provider on the protocol page.

Trusted Providers			
Name	Enabled	Metadata Expiration Date	Metadata Repository
<b>Identity Providers</b>			
<input type="checkbox"/> ipd-saml2-206	<input checked="" type="checkbox"/>	Not specified	
<b>Service Providers</b>			
No items			

- 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 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 216](#).

## 7.3.2 Creating a Trusted Service Provider for SAML 1.1 and Liberty

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 1.4.4, “Managing the Keys, Certificates, and Trust Stores,” on page 31](#).
- ♦ 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 7.9.4, “Editing a SAML 1.1 Service Provider’s Metadata,” on page 229](#).

- ♦ 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 or Liberty**.
- 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 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:** Paste the 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 229](#).

**Metadata Repositories:** Allows you to configure several identity and/or service providers using a multi-entity metadata file available in a central repository. For more information about creating Identity and/or Service Providers see, [Section 7.3.4, “Creating Identity Providers and Service Providers,” on page 215](#).

- 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 216](#).

## 7.3.3 Creating a Trusted Identity Provider

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 1.4.4, "Managing the Keys, Certificates, and Trust Stores," on page 31](#).
- Shared the trusted root of the SSL certificate of your Identity Server with the identity provider so that the administrator can import 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 7.9.3, "Editing a SAML 1.1 Identity Provider's Metadata," on page 228](#).
- 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.

The `cacerts` file is located at:

**Linux:** `/opt/novell/java/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 228](#).

**Metadata Repositories:** Allows you to configure several identity and/or service providers using a multi-entity metadata file available in a central repository. For more information about creating Identity and/or Service Providers see, [Section 7.3.4, “Creating Identity Providers and Service Providers,” on page 215.](#)

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:

---

**NOTE:** The PID in the login URL must exactly match the entity ID specified in the metadata.

---

```
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 251.](#)

**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 216.](#)

## 7.3.4 Creating Identity Providers and Service Providers

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

For the protocol, click **SAML 1.1** or **SAML 2.0**.

2 Click **New**, then click **Identity Provider** or **Service Provider**.

3 Select **Metadata Repositories** from the **Source** drop down list and select the repository name from the **Repository** field.

4 Select the entities to add SAML 1.1 or SAML 2.0 as Identity or Service Providers and click **Finish**.

The entities that are already assigned to the cluster have the details displayed in the **Assigned to Cluster** column.



The default settings of identity and service providers when you import the metadata repository are as follows:

- ♦ *SAML 1.1 Identity Provider*
  - ♦ No contracts associated to Satisfiable list of IDP
  - ♦ No image selected for the IDP card
  - ♦ Login URL will be empty with Show card disabled.
  - ♦ No attribute set associated
- ♦ *SAML 1.1 Service Provider*
  - ♦ No contracts associated to Satisfiable list of SP
  - ♦ No Attribute set associated
- ♦ *SAML 2.0 Identity Provider*
  - ♦ Persistent Federation as the Name Identifier
  - ♦ Post Binding
  - ♦ No contracts associated to Satisfiable list of IDP
  - ♦ No image selected for the IDP card
  - ♦ No Attribute set associated
- ♦ *SAML 2.0 Service Provider*
  - ♦ No contracts associated to Satisfiable list of SP
  - ♦ Post Binding
  - ♦ No Attribute set associated

## 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 7.7, “Configuring Communication Security,” on page 220.](#)
- ♦ **Attributes to Obtain at Authentication:** See [Section 7.8.1, “Configuring the Attributes Obtained at Authentication,” on page 224.](#)
- ♦ **Metadata:** See [Section 7.9, “Managing Metadata,” on page 227.](#)
- ♦ **Authentication Request:** See [Section 7.10, “Configuring an Authentication Request for an Identity Provider,” on page 232.](#)
- ♦ **User Identification:** See [Chapter 13, “Configuring User Identification Methods for Federation,” on page 369.](#)
- ♦ **Authentication Card:** See [Section 7.17, “Managing the Authentication Card of an Identity Provider,” on page 247.](#)

You can modify the following features of a service provider:

- ♦ **Communication Security:** See [Section 7.7, “Configuring Communication Security,” on page 220.](#)



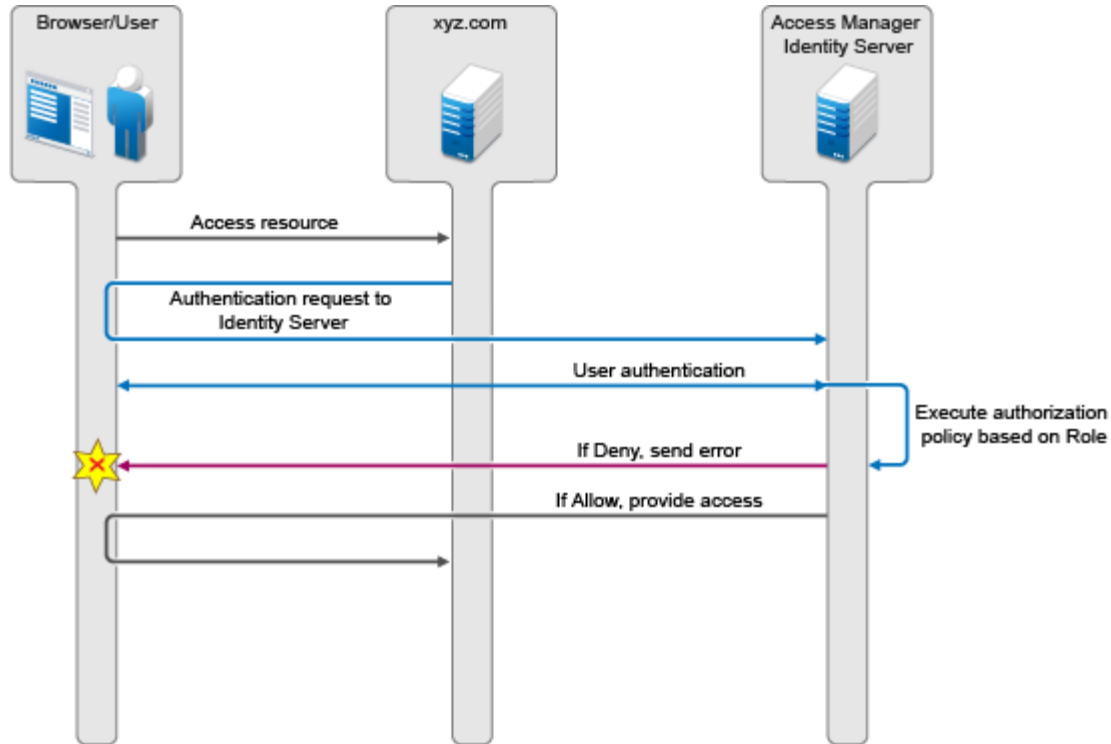
- ♦ **Attributes to Send in the Response:** See [Section 7.8.2, “Configuring the Attributes Sent with Authentication,”](#) on page 225.
- ♦ **Intersite Transfer Service:** See [“Configuring an Intersite Transfer Service Target for a Service Provider”](#) on page 254.
- ♦ **Metadata:** See [Section 7.9, “Managing Metadata,”](#) on page 227.
- ♦ **Authentication Response:** See [Section 7.11, “Configuring an Authentication Response for a Service Provider,”](#) on page 237.

## 7.5 Executing Authorization Based Roles Policy During SAML 2.0 Service Provider Initiated Request

Access Manager service provider federation profiles do not currently allow control based on authorization policies. Usually the service providers enforce authorization rules. However, not all service providers have this flexibility. It is recommended not to trust the service provider to enforce such rules. You can now apply an authorization policy to a configured service provider to either allow or not to allow access to the service provider. The Identity Server will evaluate the service provider and will generate the successful/unsuccessful assertions.

**Use Case:** Company xyz.com uses a CRM application that is protected through the SAML 2.0 service provider. This application should only be accessible to the sales team and not to any other teams. Whenever a user accesses the application through the service provider, it redirects to the Identity Server for validating the user.

*Figure 7-3 Executing Authorization Policy Based on Role*



The Identity Server should authenticate the user and then check if the user is member of the sales team. If the user is a member, then the Identity Server sends a successful assertion to the service provider. Else, the Identity Server sends an error response to the service provider.

By default, the Identity Server executes these authorization policies after a user is authenticated during spsend. Adding the `ALLOW_AUTH_POLICY_EXECUTION=false` option in the `nidp.properties` file will not allow the Identity Server to execute the authorization policies.

If the authorization policy is to deny execution, the Identity Server sends the following message as part of an assertion response. `<samlp:Status>`

```
<samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Responder">
    <samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:RequestDenied" />
    </samlp:StatusCode>
    <StatusMessage>Authorization is failed</StatusMessage>
</samlp:Status>
```

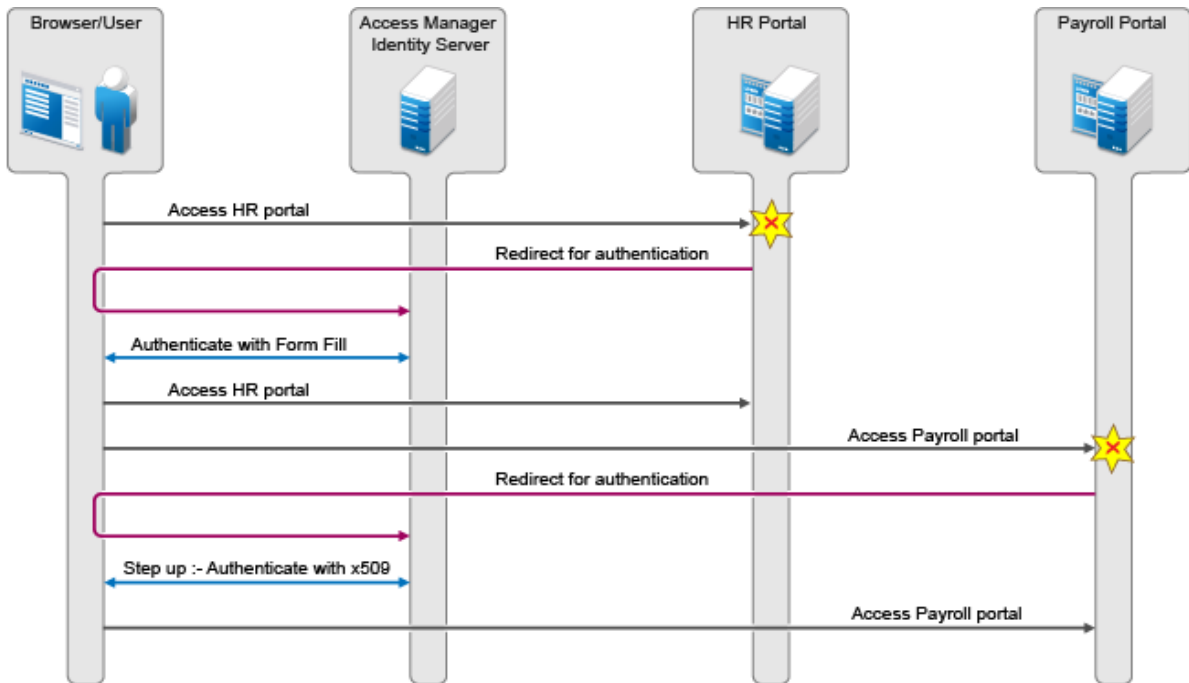
For more information about, configuring a brokering for authorization of service providers see [Section 12.3, “Configuring a Brokering for Authorization of Service Providers,” on page 354.](#)

## 7.6 Contracts Assigned to SAML 2.0 Service Provider

During federation, when a service provider initiates an authentication request, contract information may not be available. If the contract information is not available, the Identity Server executes a default contract for validating the user. The step up authentication feature enables you to assign a default contract for service providers in such scenarios.

The following scenario helps you understand the execution of contracts that are assigned to the SAML 2.0 service provider.

**Figure 7-4** Step Up Authentication example with two applications:



There are two applications Payroll and HR web applications protected through different service providers and are using Access Manager Identity Server as identity provider. The user wants to use the name/password form contract whenever the user accesses the HR application and wants to use the higher level contract say X509 for the Payroll application. The Identity Server provides ability to execute the appropriate contract that has been assigned to the service provider instead of executing the default contract.

The following procedure allows you to assign a specific contract to the service provider.

- 1 Click on **Devices > Identity Servers > Edit > SAML2.0**.
- 2 Click on configured service provider.
- 3 Go to **Options > Step Up Authentication** contracts and select the contracts from the **Available contracts** list.

The following table lists the behavior of a service provider request.

Service Provider Request	Result (Identity Server Response if the user is not authenticated)
<b>Service provider request has no contract information to be executed at the Identity Server.</b>	
1. Identity Server has no contracts set for this service provider as in <a href="#">Step 3</a> .	Execute default contract for validating the user and default contract name will be sent in the response.
2. Identity Server has contract C1 set for this service provider as in <a href="#">Step 3</a> .	C1 will be executed for validating the user and C1 will be sent in response.
<b>Service provider requests execution of contract C1 at the Identity Server.</b>	
1. Identity Server has no contracts set for this service provider as in <a href="#">Step 3</a> .	C1 will be executed for validating the user and C1 will be sent in response.

Service Provider Request	Result (Identity Server Response if the user is not authenticated)
2. Identity Server has contract C1 set for this service provider as in <a href="#">Step 3</a> .	C1 will be executed for validating the user and C1 will be sent in response.
3. Identity Server has contract C2 set for this service provider. C2 has trust level check disabled.	C2 will be executed for validating the user and C2 will be sent in response. ( <b>Note:</b> This is as good as C1 not available at the Identity Server)
4. Identity Server has contract C2 set for this service provider. C2 has trust level check enabled.	<p>If trust level of C2 <math>\geq</math> trust level of C1, then C2 will be executed and C2 will be sent in response.</p> <p>If trust level of C2 <math>&lt;</math> trust level of C1, then C1 will be executed and C1 will be sent in response as in the previous Access Manager 3.2 release.</p> <p>If C1 is not available at Identity Server, then C2 is executed and C2 is sent in the response.</p>

## 7.7 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 about replacing the default certificate, see [Section 1.4.4, “Managing the Keys, Certificates, and Trust Stores,” on page 31](#).

**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 1.4.4, “Managing the Keys, Certificates, and Trust Stores,” on page 31](#).

**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 7.7.1, “Configuring Communication Security for Liberty and SAML 1.1,” on page 221](#)
- [Section 7.7.2, “Configuring Communication Security for a SAML 2.0 Identity Provider,” on page 222](#)
- [Section 7.7.3, “Configuring Communication Security for a SAML 2.0 Service Provider,” on page 223](#)

## 7.7.1 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.

## 7.7.2 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.

Configuration Metadata Authentication Card

Trust Attributes User Identification Options

Name: ipd-saml2-206

**Security**

☐ Encrypt name identifiers

**SOAP Back Channel Security Method**

☒ Message Signing

☐ Mutual SSL

☐ Basic Authentication

**Send:**

Name:

Password:

**Verify:**

Name:

Password:

Certificate Revocation Check Periodicity: On Startup  Hourly

- 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.

**Certificate Revocation Check Periodicity:** Specifies if the certificate is valid or not. You can define periodicity to validate on start up, on assertion level, or set frequency to hourly/daily.

- 4 Click **OK** twice.
- 5 Update the Identity Server.

## 7.7.3 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.

**Certificate Settings:** All service providers use the default signing and encryption certificates of the identity provider. Specify custom certificates for a service provider as follows:

- ♦ **Identity Provider Signing Certificate:** Select a certificate from the keystore and assign it to the service provider.
- ♦ **Identity Provider Encryption Certificate:** Select a certificate from the keystore and assign it to the service provider.

---

**NOTE:** When you assign custom certificates to each service provider while configuring the identity server, ensure that you export these certificates and custom metadata to the service provider. To retrieve the metadata, click on the metadata link. (This link is available in the note on the **Trust** page).

For example, the default certificates will have the following default metadata URL: `<IDP URL>/nidp/saml2/metadata`.

The custom certificates will have the following custom metadata URL for a service provider: `<IDP URL>/nidp/saml2/metadata?PID=<SP Entity ID>`

- 
- ♦ **Certificate Revocation Check Periodicity:** Specifies if the certificate is valid or not. You can define periodicity to validate on start up, on assertion level, or set frequency to hourly/daily.

**SOAP Back Channel Security Method:** 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.

## 7.8 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 191](#)).

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 7.8.1, "Configuring the Attributes Obtained at Authentication," on page 224](#)
- ♦ [Section 7.8.2, "Configuring the Attributes Sent with Authentication," on page 225](#)
- ♦ [Section 7.8.3, "Sending Attributes to the Embedded Service Provider," on page 226](#)

### 7.8.1 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 about this process, see [Section 6.1, “Configuring Attribute Sets,”](#) on [page 191](#).

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.

## 7.8.2 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 about this process, see [Section 6.1, “Configuring Attribute Sets,” on page 191](#).

**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.

### 7.8.3 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.

## 7.9 Managing Metadata

The Liberty, SAML 1.1, and SAML 2.0 protocols contain pages for viewing and reimporting the metadata of the trusted providers.

- ♦ [Section 7.9.1, “Viewing and Reimporting a Trusted Provider’s Metadata,” on page 227](#)
- ♦ [Section 7.9.2, “Viewing Trusted Provider Certificates,” on page 228](#)
- ♦ [Section 7.9.3, “Editing a SAML 1.1 Identity Provider’s Metadata,” on page 228](#)
- ♦ [Section 7.9.4, “Editing a SAML 1.1 Service Provider’s Metadata,” on page 229](#)
- ♦ [Section 7.9.5, “Editing a SAML 2.0 Service Provider’s Metadata,” on page 231](#)

### 7.9.1 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.

---

**NOTE:** The trusted providers that are from the metadata repository cannot be reimported from this option. Go to **Shared Settings > Metadata Repositories** and click on the metadata repository created to reimport the trusted provider.

---

- 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.

---

**NOTE:** Reimport support is not available for SAML 1.1 and SAML 2.0 protocols.

---

## 7.9.2 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.

## 7.9.3 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**.

Supported version:

**Provider ID:**

Source ID:

Metadata expiration:

SAML attribute query URL:

Artifact resolution URL:

**Signing Certificates**

Attribute authority:

Identity provider:

- 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.

## 7.9.4 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 481](#).

- 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**.

The screenshot shows the 'Configuration' tab with the 'Metadata' sub-tab selected. The 'View' button is disabled, and the 'Edit' button is active. The 'Certificates' button is also visible. The 'Supported version' dropdown is set to 'SAML 1.0 and SAML 1.1'. The 'Provider ID' field contains the URL 'https://jwilson.provo.novell.com:8443/nidp/saml/metad'. The 'Metadata expiration' field is empty, and the 'Want Assertion to be signed' checkbox is unchecked. The 'Artifact consumer URL' field contains 'https://jwilson.provo.novell.com:8443/nidp/saml/spass', and the 'Post Consumer URL' field contains 'https://jwilson.provo.novell.com:8443/nidp/saml/spass'. Below these fields is a 'Signing Certificate' section with a 'Service provider' field and a 'Browse...' button.

- 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**.

## 7.9.5 Editing a SAML 2.0 Service Provider's Metadata

Access Manager allows you to obtain metadata for SAML 2.0 providers. However, metadata for SAML 2.0 might not be available for some service 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 481.

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > SAML 2.0 > [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**.

The screenshot shows the 'Edit Metadata' page in the Administration Console. The 'Metadata' tab is active. The 'Provider ID' field contains the URL 'https://idp245.idpdomain.com:8443/nidp/saml2/metadata'. The 'Metadata expiration' field is empty, and the 'Want Assertion to be signed' checkbox is unchecked. The 'Artifact consumer URL' field is empty, and the 'Post consumer URL' field contains 'https://idp245.idpdomain.com:8443/nidp/saml2/spasse'. The 'Signing Certificate' section at the bottom has a 'Service provider' field and a 'Browse...' button.

- 4 Fill in the following fields:

**Provider ID:** (Required) Specifies the SAML 2.0 metadata unique identifier for the provider. For example, `https://<dns>:8443/nidp/saml2/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/saml2/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/saml2/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**.

## 7.10 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 7.10.1, “Configuring a Liberty Authentication Request,” on page 232](#)
- ♦ [Section 7.10.2, “Configuring a SAML 2.0 Authentication Request,” on page 234](#)

### 7.10.1 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, ensure that you configure a user identification method. See [Section 13.1.1, “Selecting a User Identification Method for Liberty or SAML 2.0,” on page 369](#).

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 128](#).

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.

## 7.10.2 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:** A persistent identifier federates the user profile on the identity provider with the user profile on the service provider. It remains intact between sessions.

The persistent identifier is saved to the user data store and hides the user's identity to prevent tracking of user activities across different relying parties.

- ♦ **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, ensure that you configure a user identification method. See [Section 13.1.1, "Selecting a User Identification Method for Liberty or SAML 2.0," on page 369](#).

**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 236](#)):

- ♦ **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 236](#)):

- ♦ **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 128](#).

Most third-party identity providers do not support contracts.

#### 4 Configure the options:

**Response protocol binding:** Select **Artifact** or **Post** or **Let IDP Decide**. Artifact and Post are the two methods for transmitting assertions between the authenticating system and the target system.

If you select **Let IDP Decide**, the binding is selected based on the profile that is enabled at Identity Provider and the binding selected in the service provider.

---

**NOTE:** The post binding can be configured to be sent as a compressed option. Perform the following steps to achieve this:

1. Open the `nidpconfig.properties` file located in `/opt/novell/nam/idp/webapps/nidp/WEB-INF/classes`.
2. Modify the following:  
  
**SAML2\_POST\_DEFLATE\_TRUSTEDPROVIDERS:** Enter trusted provider's name, metadata URI, or provider ID. You can specify multiple trusted providers in a comma separated format. These are the trusted providers who expect SAML2 POST messages in deflated format. In other words, this provider has to send deflated SAML2 POST messages to the listed trusted providers.  
  
**IS\_SAML2\_POST\_INFLATE:** Specify `True`, if this provider will receive deflated SAML2 POST messages from its trusted providers.
3. Restart the Identity Server by using this command: `/etc/init.d/novell-idp restart`.

---

**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 request 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 NetIQ 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 NetIQ 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 NetIQ 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.

## 7.11 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 7.18, “Using the Intersite Transfer Service,” on page 248](#)) 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 7.11.1, “Configuring the Liberty Authentication Response,” on page 237](#)
- ♦ [Section 7.11.2, “Configuring the SAML 2.0 Authentication Response,” on page 239](#)
- ♦ [Section 7.11.3, “Configuring the SAML 1.1 Authentication Response,” on page 241](#)

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 137](#).

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 about how to configure such a contract, see [Section 3.5.2, “Creating a Contract for a Specific Authentication Type,” on page 138](#).

### 7.11.1 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**.

Supported identity formats	Use	Default
Persistent identifier format:	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>
Transient identifier format:	<input checked="" type="checkbox"/>	<input type="radio"/>

☒ Use proxied requests

☒ Provide Discovery Services

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 206.

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 a persistent identifier that federates the user profile on the identity provider with the user profile on the service provider. It remains intact between sessions.
- ♦ **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.

## 7.11.2 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**.

The screenshot shows the 'Authentication Response' configuration page in the Administration Console. The page has tabs for 'Configuration' and 'Metadata', with 'Configuration' selected. Below the tabs are links for 'Trust', 'Attributes', 'Authentication Response', and 'Intersite Transfer Service'. The 'Binding' dropdown is set to 'Artifact'. A table lists various identifier formats with checkboxes and default values. Below the table are checkboxes for 'Use proxied requests' and 'Include the Session Timeout attribute in the assertion'. At the bottom, 'Assertion Validity' is set to 300 seconds.

Name	Identifier	Format	Default	Value
<input checked="" type="checkbox"/>	Persistent	<input type="radio"/>	Automatically generated	
<input checked="" type="checkbox"/>	Transient	<input checked="" type="radio"/>	Automatically generated	
<input type="checkbox"/>	E-mail	<input type="radio"/>	<Not Specified>	
<input type="checkbox"/>	Kerberos	<input type="radio"/>	<Not Specified>	
<input type="checkbox"/>	X509	<input type="radio"/>	<Not Specified>	
<input type="checkbox"/>	Unspecified	<input type="radio"/>	<Not Specified>	

☒ Use proxied requests

☐ Include the Session Timeout attribute in the assertion

Assertion Validity  seconds

- 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 206.

---

**NOTE:** The post binding can be configured to be sent as a compressed option. Perform the following steps to achieve this:

1. Open the `nidpconfig.properties` file located in `/opt/novell/nam/idp/webapps/nidp/WEB-INF/classes`.
  2. Modify the following:  
**SAML2\_POST\_DEFLATE\_TRUSTEDPROVIDERS:** Enter trusted provider's name, metadata URI, or provider ID. You can specify multiple trusted providers in a comma separated format. These are the trusted providers who expect SAML2 POST messages in deflated format. In other words, this provider has to send deflated SAML2 POST messages to the listed trusted providers.  
**IS\_SAML2\_POST\_INFLATE:** Specify `True`, if this provider will receive deflated SAML2 POST messages from its trusted providers.
  3. Restart the Identity Server by using this command: `/etc/init.d/novell-idp restart`.
-



- 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 369](#).

---

- 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 7.8.1, “Configuring the Attributes Obtained at Authentication,” on page 224](#)). 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 When the **Include the Session Timeout attribute in the assertion** option is enabled, the Identity Server sends the Identity Server session time out value to the service provider in the assertion.
- 8 You can manually set the assertion validity time for the SAML service provider in the **Assertion Validity** field to accommodate clock skew between the service provider and SAML Identity Server (IDP).
- 9 Click **OK** twice, then update the Identity Server.



## 7.11.3 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**.

The screenshot shows the 'Authentication Response' configuration page. At the top, there are tabs for 'Configuration' and 'Metadata'. Below these are sub-tabs: 'Trust', 'Attributes', 'Authentication Response' (which is active), and 'Intersite Transfer Service'. The main content area is divided into two sections. The first section, 'Name Identifier Format', has a table with three rows: 'Unspecified' (selected with a radio button), 'E-mail', and 'X509'. Each row has a corresponding dropdown menu for the 'Value' column, all of which currently show '<Not Specified>'. The second section, 'Audiences', has a 'New | Delete' link and a checkbox labeled 'Audience'. Below this, there is a text input field containing the URL 'https://sp242.labs.blr.novell.com:8443/nidp/saml/metadata'. At the bottom, there is a field for 'Assertion Validity' set to '300' seconds.

- 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 You can manually set the assertion validity time for the SAML service provider in the **Assertion Validity** field to accommodate clock skew between the service provider and SAML Identity Server (IDP).
- 7 Click **OK** twice, then update the Identity Server.

## 7.12 Routing to an External Identity Provider Automatically

When the NetIQ Identity Server is configured to federate with multiple external Identity Providers, administrator can specify the list of Authentication Contracts that an external provider can execute. This configuration allows the NetIQ Identity Server (acting as service provider) to automatically select the external identity provider without the user having to click on the external provider's card.

Authentication Contracts in the NetIQ Identity Servers have been enhanced to be configured with an Authentication Class Reference. This reference can be used in federating with External Identity or Service Providers that only respond to `AuthnContextClassRef` in the Authentication Request and Response. For more information about setting up the contract mapping and adding contracts to the satisfiable list, see [Section 7.17.1, “Modifying the Authentication Card for Liberty or SAML 2.0,” on page 247](#) and [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).

## 7.13 Defining Options for Liberty or SAML 2.0

According to the *Single Logout Profile* in *OASIS SAML V2.0* profiles, session users should use a front channel binding. This profile is initiated to maximize the likelihood that the session authority can successfully propagate the logout to all users.

### 7.13.1 Defining Options for SAML 2.0 Identity Provider

- 1 In Administration Console, click **Devices > Identity Servers > Servers > Edit > SAML 2.0 > Identity Provider > Options**.

The screenshot shows the 'Options' tab in the NetIQ Identity Server Administration Console. The 'Options' tab is selected, and the 'OIOSAML Compliance' and 'Enable front channel logout' checkboxes are visible. Below these are links for 'New' and 'Delete', and a table with columns 'Name' and 'Value' showing 'No items'. At the bottom are 'OK', 'Cancel', and 'Apply' buttons.

- 2 **OIOSAML Compliance:** Enable this option to make Identity Provider OIOSAML compliant.
- 3 **Enable Front Channel Logout:** After this option is enabled, the Service Provider initiates a logout at the Identity Provider by using the HTTP Redirect method.

## 7.13.2 Defining Options for Liberty or SAML 2.0 Service Provider

NetIQ Access Manager can be used as an identity provider for several service providers. You can configure a specific authentication contract that is required for a Service provider. If more than one authentication contract is configured for a service provider, the contract having minimum level will be selected.

When providing authentication to a service provider, the identity server ensures that the user is authenticated by the required contract. When a user is not authenticated or when user is authenticated, but the authenticated contracts do not satisfy the required contracts, user will be prompted to authenticate with required contract. This is called step up authentication.

If no required contract is configured, then the default contract is executed.

---

**NOTE:** This step up authentication is supported only for Intersite Transfer Service (identity provider initiated) requests on Liberty and works for both identity and service provider initiated requests for SAML 2.0.

---

### To Define Options for Liberty Service Provider

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

The screenshot shows the 'Options' tab in the NetIQ Administration Console. The 'Step Up Authentication contracts' section contains two lists: 'Selected contracts' and 'Available contracts'. The 'Available contracts' list includes 'Name/Password - Basic', 'Secure Name/Password - Basic', 'Secure Name/Password - Form', and 'Name/Password - Form'. Arrows between the lists allow moving contracts. At the bottom are 'OK', 'Cancel', and 'Apply' buttons.

- 2 Select the required step up authentication contracts from the **Available contracts** list and move them to the **Selected contracts** list. This is to provide the step up authentication for the service provider.
- 3 Click **OK**.

### To Define Options for SAML 2.0 Service Provider

- 1 In Administration Console, click **Devices > Identity Servers > Servers > Edit > SAML 2.0 > Service Provider > Options**.

- 2 **OIOSAML Compliance:** Enable this option to make the service provider OIOSAML compliant. The OIOSAML attribute set is automatically populated with the required attributes to send with authentication after selecting this check box.
- 3 Select the required step up authentication contracts from the **Available contracts** list and move them to the **Selected contracts** list. This is to provide the step up authentication for the service provider.

---

**NOTE:** The contract that is configured first in the **Selected contracts** list will only be executed. This is applicable only for SAML 2.0.

---

- 4 Click **OK**.

### 7.13.3 Defining Options for Liberty Identity Provider

- 1 In Administration Console, click **Devices > Identity Servers > Servers > Edit > Liberty or SAML 2.0 > Identity Provider > Options**.

- 2 **Enable Front Channel Logout:** After this option is enabled, Service Provider initiates a logout at the Identity Provider by using the HTTP Redirect method.
- 3 **Configure Front Channel Logout for Access Gateway Initiated Logout:** In addition to enabling the front channel logout, add the following parameters at the NESP web.xml and restart tomcat:

Add the following parameters in the web.xml below the IdapLoadThreshold context param:

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

```
</context-param>
```

To restart tomcat:

Linux: Enter the following command: `/etc/init.d/novell-idp restart`

Windows: Enter the following commands:

`net stop Tomcat7`

`net start Tomcat7`

## 7.14 Defining Options for SAML 1.1 Service Provider

For more information about Options, see [Section 7.13.2, “Defining Options for Liberty or SAML 2.0 Service Provider,” on page 243](#)

---

**NOTE:** Step up authentication is supported only for identity provider initiated requests on SAML 1.1.

---

- 1 In Administration Console, click **Devices > Identity Servers > Servers > Edit > SAML 1.1 > Service Provider > Options**.

The screenshot shows the 'Options' tab in the Administration Console for a SAML 1.1 Service Provider. The window has a title bar with 'Configuration' and 'Metadata' tabs. Below the title bar is a navigation bar with 'Trust', 'Attributes', 'Authentication Response', 'Intersite Transfer Service', and 'Options'. The main content area is titled 'Step Up Authentication contracts'. It contains two lists: 'Selected contracts:' on the left and 'Available contracts:' on the right. The 'Available contracts' list includes 'Name/Password - Basic', 'Secure Name/Password - Basic', 'Secure Name/Password - Form', and 'Name/Password - Form'. There are arrows between the lists to move items. At the bottom of the window are 'OK', 'Cancel', and 'Apply' buttons.

- 2 Select the required step up authentication contracts from the **Available Contracts** list and move them to the **Selected Contracts** list. These selected contracts will be used to provide the step up authentication for the service provider.
- 3 Click **OK**.

## 7.15 Defining Session Synchronization for the A-Select SAML 2.0 Identity Provider

If a user session is active on the Service Provider, the service provider periodically sends session synchronization to the Identity Server to maintain the session. You must configure the properties for the session synchronization between the service provider and the target Identity provider.

- 1 In the Administration Console, click **Devices > Identity Servers > Servers > Edit > Liberty or SAML 2.0 > Identity Provider > Options**.

Name	Value
config.aselect.sessionsync.enabled	true
config.aselect.sessionsync.url	https://test.federatie.overheid.nl/aselectserver/server/saml20_idp_session_sync

- 2 Click **New > Add Properties**, then specify the following values:

**Property Name:** Specify **config.aselect.sessionsync.enabled**

**Property Value:** Specify **true**.

- 3 For session synchronization, add two options, one to enable the session synchronization and the other to provide the URL to which synchronization message should be sent.

The session synchronization message is sent from the Access Manager Service Provider to the A-Select Identity Provider, in tandem with the Access Gateway ESP's activity update. The session synchronization message is sent only if the user session is active at the Access Gateway portal, which is the ESP to the Access Manager Service Provider. If you log in directly to the Access Manager Service Provider, even if the session is active, the session synchronization message is not sent to the A-Select Identity Provider.

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

## 7.16 Configuring the Liberty or SAML 2.0 Session Timeout

When you are active on a session on the service provider and a timeout occurs, the service provider initiates a logout. You can configure this timeout by using the `web.xml` parameter in the Access Gateway ESP, then ESP initiates a logout message to the Access Manager Service Provider over the SOAP back channel when the timeout is reached. After the Service Provider receives this message, it creates a SAML 2.0 logout request to the remote Identity Provider over SOAP.

To send session timeout message:

- 1 Open the `web.xml` file located at:

**Linux:** `/opt/novell/nam/idp/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>notifysessionTimetoIDP</param-name>
    <param-value>true</param-value>
</context-param>
```

ESP will send a ESP session timeout message then on timeout, the service provider will send a `samlp:LogoutRequest` `xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"` request to the remote Identity provider.

- 3 Save the file, then copy it to each Identity Server in the cluster.
- 4 Restart Tomcat on each Identity Server in the cluster using the following command:

**Linux:** Enter the following command:

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

**Windows:** Enter the following commands:

```
net stop Tomcat7
```

```
net start Tomcat7
```

## 7.16.1 Session Termination

If you set the session synchronization between the Service Provider and remote Identity Provider, then the remote Identity Provider never sends the logout request to the active Service Provider.

## 7.17 Managing the Authentication Card of an Identity Provider

- ♦ [Section 7.17.1, “Modifying the Authentication Card for Liberty or SAML 2.0,” on page 247](#)
- ♦ [Section 7.17.2, “Modifying the Authentication Card for SAML 1.1,” on page 248](#)

### 7.17.1 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.

---

**NOTE:** Do not disable the **Show Card** option for default contracts.

---

**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.

**Satisfies Contract:** Select the required contracts from the **Available contracts** list and move them to the **Satisfies contract** list. This creates a mapping between external provider class reference to local authentication contracts.

- 3 Click **OK** twice, then update the Identity Server.



## 7.17.2 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.

---

**NOTE:** The PID in the login URL must exactly match the entity ID specified in the metadata.

---

```
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 251](#).

If your identity provider is a NetIQ 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>
```

```
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 254](#).

**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.

## 7.18 Using the Intersite Transfer Service

- [Section 7.18.1, “Understanding the Intersite Transfer Service URL,” on page 249](#)
- [Section 7.18.2, “Specifying the Intersite Transfer Service URL for the Login URL Option,” on page 251](#)
- [Section 7.18.3, “Using Intersite Transfer Service Links on Web Pages,” on page 252](#)
- [Section 7.18.4, “Configuring an Intersite Transfer Service Target for a Service Provider,” on page 254](#)



- [Section 7.18.5, “Configuring Whitelist of Target URLs,” on page 255](#)
- [Section 7.18.6, “Federation Entries Management,” on page 255](#)
- [Section 7.18.7, “Step up Authentication Example for an Identity Provider Initiated Single Sign-On Request,” on page 255](#)

## 7.18.1 Understanding the Intersite Transfer Service URL

The Intersite Transfer Service is used by an identity provider to provide authentication to occur at a service provider that it trusts. The URLs for accessing the Intersite Transfer Service differ for each supported protocol (Liberty, SAML 1.1, and SAML 2.0). The NetIQ Access Manager identity and service provider components use the following format of the Intersite Transfer Service URL:

`<identity_provider_URL>?PID=<entityID>&TARGET=<final_destination_URL>`

The `<identity_provider_URL>` is the location where the authentication request can be processed. For an Access Manager Identity Server, the URL is the Base URL of the server that provides 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 provides the authentication, refer the 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_provider_URL>`.

For SAML 1.1 and SAML 2.0, search the metadata for its entityID value. For Liberty, search the metadata for the providerID value. Access Manager 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, refer the 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`), the URL must be encoded to prevent it from being truncated.

For example:

- **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`

To read more about configuring an intersite URL, see [“Configuring an Intersite Transfer Service Target for a Service Provider” on page 254](#).

If you configure an Intersite Transfer Service URL for an Identity Server that is the Access Manager Identity Server 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_provider_URL>?id=<user_definedID>`

For example:

- ♦ **SAML 2.0:** `https://test.blr.novell.com:8443/nidp/saml2/idpsend?id=testssaml2&TARGET=https://eng.provo.novell.com`
- ♦ **SAML 1.1:** `https://testsb.blr.novell.com:8443/nidp/saml/idpsend?id=testssaml&TARGET=https://eng.provo.novell.com`
- ♦ **Liberty:** `https://testsb.blr.novell.com:8443/nidp/idff/idpsend?id=libertytest&TARGET=https://eng.provo.novell.com`

If the **Allow any target** option is enabled and if the Intersite Transfer Service URL has a target value, then the user is redirected to target URL.

The Intersite Transfer Service URL for SAML 2.0 will be `https://testsb.blr.novell.com:8443/nidp/saml2/idpsend?id=testssaml2&TARGET=http://www.google.com` where `http://www.google.com` is the target URL.

---

**NOTE:** Depending on the usage, the target parameter serves different purpose. It is case-sensitive.

- ♦ **target:** Specifies the idpsend URL with a contract id.
  - ♦ **TARGET:** Specifies URL of the final destination.
- 

**Use case:** If authentication with a particular contract is enabled in Intersite URL, you are redirected to the default target URL. Use the following format: `http(s)://<$idp_host_name>/nidp/app?id=<$contract_to_be_executed>&target=http(s)://<$idp_host_name>/nidp/saml2/idpsend?id=<$saml_sp_identifier>`.

For more information, see [How to access an Identity Server Intersite Transfer URL with a specific contract](#).

---

**NOTE:** The `contract_to_be_executed` is executed by the Identity Server and is case sensitive.

---

For example, `https://www.idp.com:8443/nidp/app?id=npbasic&target=https://www.idp.com:8443/nidp/saml2/idpsend?id=serviceprovider1`.

## How it works?

In the above example, identity provider authentication is done with the contract id `npbasic`. You are now redirected to the service provider by using the `saml_sp_identifier` id (`serviceprovider1`). After authentication (if configured with persistent federation), the page redirects you to the available default target, or to the nidp login page of the service provider.

For configuration and ID information, see [“Configuring an Intersite Transfer Service Target for a Service Provider” on page 254](#).

In the Intersite Transfer Service URL, id can be used for the following purposes:

1. To simplify the intersite URL.

`<identity_provider_URL>?id=<user_definedID>`

2. To execute a particular contract in the Identity Server login service with intersite URL.

```
http(s)://<$idp_host_name/nidp/  
app?id=<$contract_to_be_executed>&target=http(s)://<$idp_host_name/nidp/saml2/  
idpsend?id=<$saml_sp_identifier>
```

## 7.18.2 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 7-5 SAML 1.1 Authentication Card*

Configuration Metadata **Authentication Card**

ID:

Text:

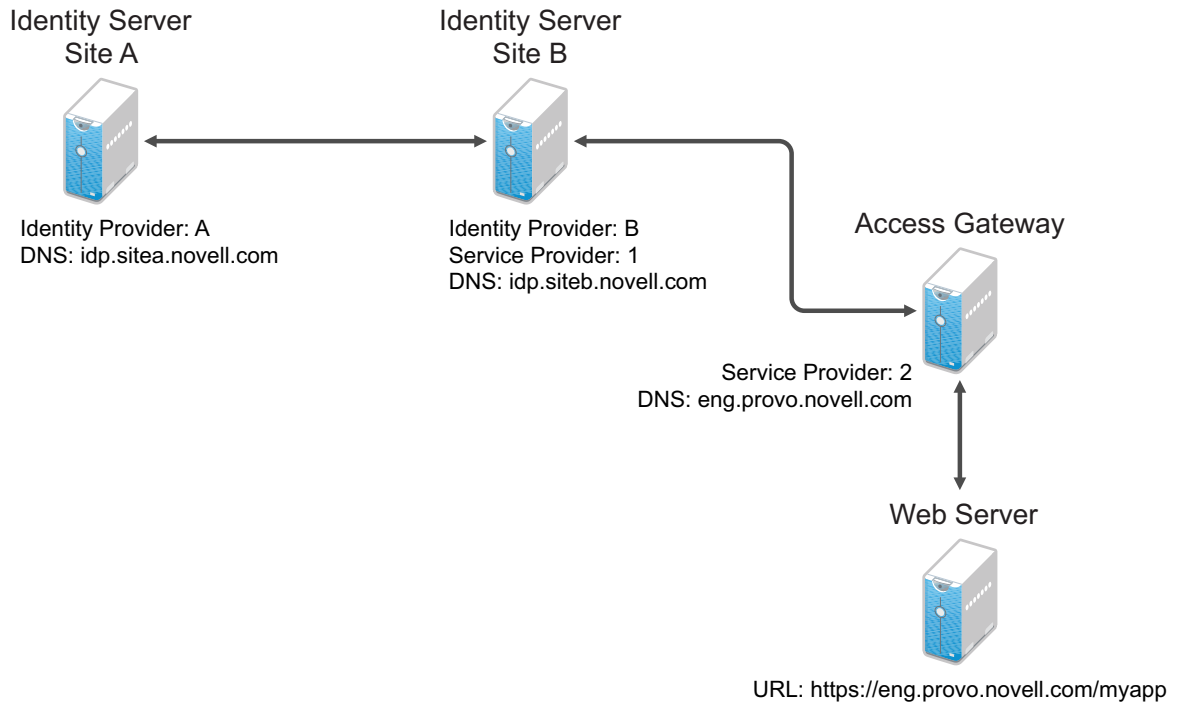
Login URL:

Image:

☒ Show Card

For a card to appear as a login option, you must specify a **Login URL** and select the **Show Card** option. [Figure 7-6](#) illustrates a possible configuration that requires the Intersite Transfer Service for the SAML 1.1 protocol.

**Figure 7-6** Federated Identity Configuration



If you want a card to appear that allows the user to log in to Site A (as shown in [Figure 7-5](#)), you need to specify a value for the **Login URL** option.

Using the DNS names from [Figure 7-6](#), 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 7.17.2, “Modifying the Authentication Card for SAML 1.1,”](#) on page 248.

### 7.18.3 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:

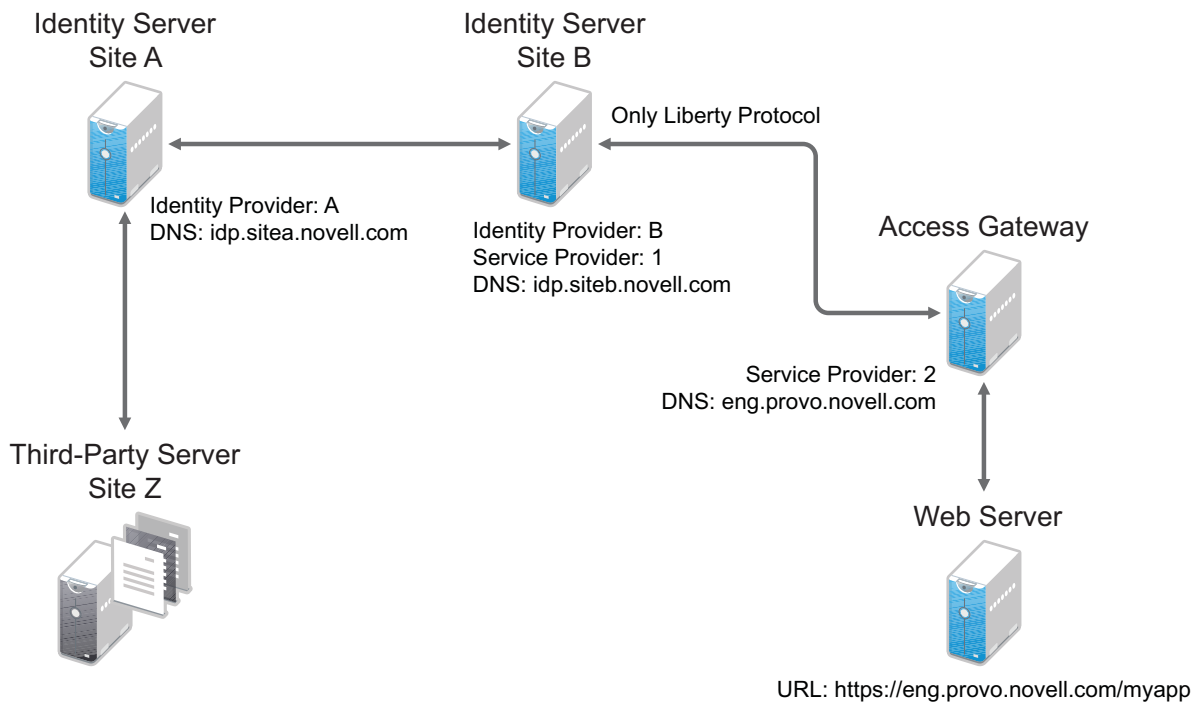
**SAML 1.1:** `<a href="https://idp.sitea.novell.com:8443/nidp/saml/idpsend?PID=https://idp.siteb.novell.com:8443/nidp/saml/metadata&TARGET=https://eng.provo.novell.com/saml1/myapp">SAML1 example</a>`

**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 7-7 illustrates a network configuration that could use these sample links.

**Figure 7-7** 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.

When defining the intersite transfer URL within the Administration Console, you can define an id and target for the SAML service provider (SP) you are accessing. For more information about accessing an Identity Server intersite transfer URL with a specific contract, see [TID 7005810](#).

## 7.18.4 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 252](#)), 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_provider_URL>?id=<user_definedID>
```

The ID specified here allows the Identity Server to find the service provider’s metadata.

**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. NetIQ recommends you to specify a default target URL, for example, `https://www.serviceprovider1.com`.

**Allow any target:** You can either select or not select this option.

- ♦ When you select this option,
  - ♦ if the Intersite Transfer URL has a target value, then the user is sent to target url.
  - ♦ if the Intersite Transfer URL does not have a target value, then the user is sent to the configured target, that is, `www.serviceprovider1.com`.
- ♦ When you do not select this option,
  - ♦ if the Intersite Transfer URL has a target value, then the user is sent to the target `www.serviceprovider1.com` irrespective of the target mentioned in the Intersite Transfer URL.
  - ♦ if the Intersite Transfer URL does not have a target value, the user is sent to `www.serviceprovider1.com`.

- 3 Click **OK** twice.

- 4 Update the Identity Server.

## 7.18.5 Configuring Whitelist of Target URLs

Redirection, which is required by many applications and services, inherently brings in a security risk. Redirects are dangerous because unsuspecting users who are visiting trusted sites can be redirected to malicious sites that exploit the users' trust. A new feature, called whitelist, has been added that restricts target URLs to specific domains.

The whitelist feature allows you to restrict target URLs to URLs which match the domains in the whitelist. Any target URLs that use a domain that is not in the list are blocked and the user receives the following error message: The request to provide authentication to a service provider has failed (outsidedomain.com-89F57BF823DFE551).

- 1 Click **Devices > Identity Servers > Edit > [Liberty, SAML1.1, or SAML 2.0] > [Service Provider] > Intersite Transfer Service**.
- 2 In the **Domain List**, click **New**.
- 3 Enter the domain name, then click **OK**.  
The domain name must be a full domain name, such as `www.novell.com`. Wildcard domain names, such as `www.novell.*.com`, do not work.
- 4 To edit an existing domain name, click the name, modify the name, then click **OK**.
- 5 To delete an existing domain name, select the check box by the domain, click **Delete**, then click **OK** to delete or **Cancel** to close the window.
- 6 Click **OK**.
- 7 Update the Identity Server.

## 7.18.6 Federation Entries Management

Identity federation is the association of accounts between an identity provider and a service provider.

## 7.18.7 Step up Authentication Example for an Identity Provider Initiated Single Sign-On Request

**Setup:** Let us assume that:

- ♦ NetIQ Access Manager is acting as an identity provider.
- ♦ The following three contracts in the identity provider are configured:
  - ♦ name password basic contract with Authentication level as 10
  - ♦ name password form contract with Authentication level as 20
  - ♦ secure name password contract with Authentication level as 30

---

**NOTE:** Enable the Satisfiable by a contract of equal or higher level option for contracts with authentication level 10 or 20 to avoid prompting for authentication when a user is already authenticated against the contract with level 30.

---

- ♦ The name password form contract for a service provider named SP\_A is configured in the identity provider.

For more information about creating and configuring the contracts, see [Section 3.4, “Configuring Authentication Contracts,” on page 128](#).

**Configuration:** Complete the following steps:

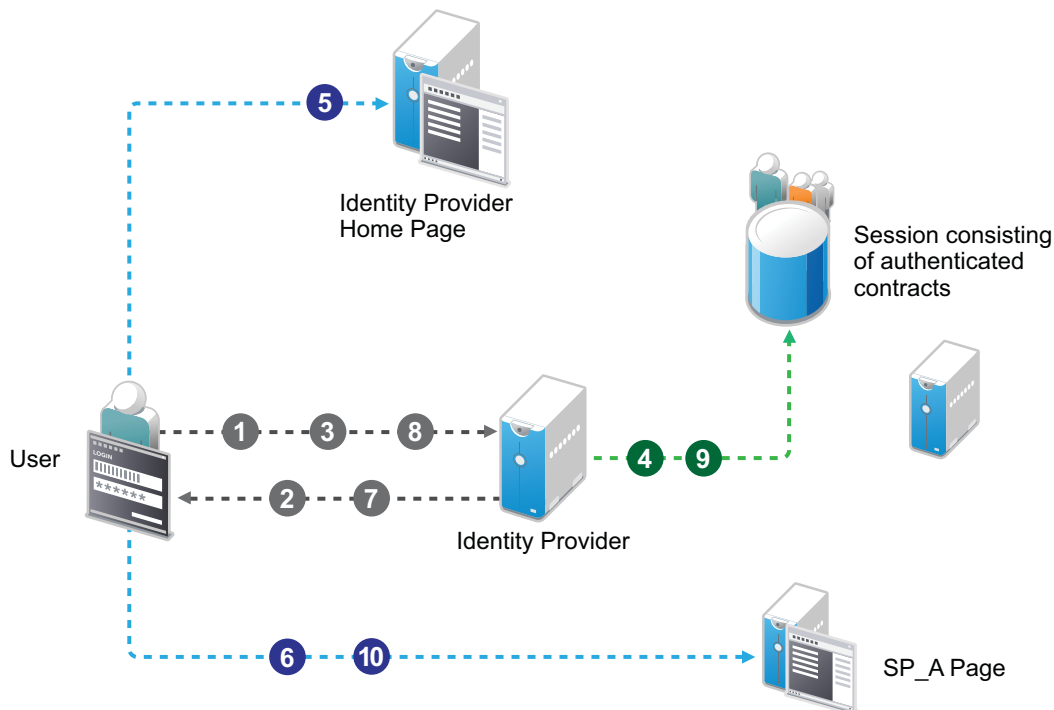
1. In the NetIQ Identity Server, configure the service provider as a trusted provider.  
For more information, see [Section 7.3, “Managing Trusted Providers,” on page 209](#).
2. In the service provider, configure the NetIQ Identity Server as a trusted provider.  
For more information, see [Section 7.3, “Managing Trusted Providers,” on page 209](#).
3. In the NetIQ Identity Server, configure the service provider with the required authentication contracts.

For information about how to configure a service provider, see [Section 7.13, “Defining Options for Liberty or SAML 2.0,” on page 242](#), [“To Define Options for Liberty Service Provider” on page 243](#) and [“Defining Options for SAML 1.1 Service Provider” on page 245](#).

**Results:** The following are the four possible scenarios:

- ♦ If the user was authenticated with the name password basic contract before making an Intersite Transfer Service request to SP\_A, the identity provider will step up to the name password form authentication.
- ♦ If the user was authenticated with the name password form contract before making an Intersite Transfer Service request to SP\_A, the identity provider will not ask for the authentication.
- ♦ If the user was authenticated with the secure name password contract before making an Intersite Transfer Service request to SP\_A, the identity provider will not ask for the authentication.
- ♦ If the user is not authenticated while making an Intersite Transfer Service request to SP\_A, the identity provider will step up to the name password form authentication.

The following diagram illustrates the workflow:





## Workflow:

- 1 User tries to authenticate in the identity provider.
- 2 User is prompted to authentication using the Name Password Basic contract.
- 3 User enters the credentials.
- 4 The Name Password Basic contract is authenticated in the identity provider and added to the user session.  
The Name Password Basic contract is the default contract in the identity provider.
- 5 User logs into the identity provider.
- 6 User makes an Intersite Transfer Service request to SP\_A.
- 7 The identity provider prompts for the authentication using the Name Password Form contract.
- 8 User enters the credentials.
- 9 The Name Password Form contract is authenticated in the identity provider and added to the user session.
- 10 User is redirected to SP\_A.

---

**NOTE:** For information about service provider initiated single sign-on and its example, see [Section 7.6, “Contracts Assigned to SAML 2.0 Service Provider,” on page 218.](#)

---

## 7.19 Enabling or Disabling SAML Tags

You can enable the following SAML tags in the Administration Console:

- ♦ **Property Name:** SAML2\_SEND\_ACS\_INDEX  
**Value:** True  
Set the value to true to send AssertionConsumerServiceIndex with AuthnRequest.
- ♦ **Property Name:** SAML2\_SEND\_ACS\_URL  
**Value:** True  
Set the value to true to send AssertionConsumerServiceURL with AuthnRequest.
- ♦ **Property Name:** Extensions  
**Value:** <samlp:Extensions><OnBehalfOf xmlns="https://idporten.difi.no/idporten-extensions">interaktor</OnBehalfOf></samlp:Extensions>  
After setting this value, Access Manager acting as a SAML 2.0 service provider makes an OnBehalfOf authentication request by using SAML extensions.
- ♦ **Property Name:** SAML\_ASSERTION\_INCLUDE\_MILLISECS  
**Property Value:** true  
Set the value to true to get SAML responses or requests including the timestamp with millisecond in IssueInstant.
- ♦ **Property Name:** SAML2\_NAMEID\_ATTRIBUTE\_NAME  
**Property Value:** ldapAttribute name  
Set the ldapattribute name to send the SAML response with the LDAP attribute value in nameidentifier.
- ♦ **Property Name:** SAML2\_AVOID\_AUDIENCE\_RESTRICTION

**Property Value:** True/False

Set the value to true to avoid sending the audience restriction information with assertion.

To enable these, perform the following steps:

- 1 In the Administration Console, click **Devices > Identity Servers > Servers > Edit > SAML 2.0**.
- 2 Based on whether the tag is for a service provider or an identity provider, select **Service Provider** or **Identity Provider** and then select **Options**.
- 3 Click **New** and specify **Property Name** and **Value**.

Enable or disable the following SAML Authentication Request tags by using the `nidpconfig.properties` file. These properties will be set at the NetIQ Access Manager Identity Server when it is configured as a SAML 2.0 service provider. Restart or wait until Access Manager refreshes the `nidpconfig.properties` file.

Property Name	Description
SAML2_AVOID_NAMEIDPOLICY	If set to true, <code>NameIDPolicy</code> is not included as part of SAML 2.0 request.
SAML2_AVOID_ISPASSIVE	If set to true, <code>IsPassive</code> is not included as part of SAML 2.0 request.
SAML2_AVOID_CONSENT	If set to true, <code>Consent</code> is not included as part of SAML 2.0 request
SAML2_AVOID_PROTOCOLBINDING	If set to true, <code>ProtocolBinding</code> is not included as part of SAML 2.0 request
SAML2_AVOID_PROXYCOUNT	If set to true, <code>ProxyCount</code> is not included as part of SAML 2.0 request
SAML2_SIGN_METHODDIGEST_SHA256	If set to true, assertion will use SHA256 algorithm as signing algorithm.
SAML2_ATTRIBUTE_CONSUMING_INDEX	<p>The value is of format <code>{SPPProviderID}-&gt;{numeric value}</code>. <code>{SPPProviderID}</code> will be replaced by the actual provider id of this service provider.</p> <p>This will set the <code>AttributeConsumingIndex</code> of SAML 2.0 requests to the numeric value specified here.</p> <p>For example, <code>https://nam.rtresearch.net:8443/nidp/saml2/metadata-&gt;2</code>.</p>
SAML2_AVOID_SPNAMEQUALIFIER	If set to true, <code>SPNameQualifier</code> is not included as part of SAML 2.0 request.
SAML2_CHANGE_ISSUER	<p>Thevalueisofformat<code>{SPPProviderID}-&gt;{issuename}</code>. <code>{SPPProviderID}</code> will be replaced by the actual provider ID of the service provider.</p> <p>This will set the issuer of SAML 2.0 requests to the issuer name specified here.</p> <p>For example, <code>https://nam.rtresearch.net:8443/nidp/saml2/metadata-&gt;https://saml.mariagerfjord.dk:8443/nidp/saml2/metadata</code>.</p>

Property Name	Description
SAML2_AVOID_SPNAMEQUALIFIER_TO	<p>Set the value to true to send SPNAMEQUALIFIER in NAMEIDENTIFIER with assertion.</p> <p>You can set this key in the <code>nidpconfig.properties</code> file in the following format:</p> <pre>https://&lt;host&gt;:&lt;port&gt;/nidp/saml2/metadata -&gt;true,https://&lt;host&gt;:&lt;port&gt;/nidp/saml2/metadata/spnameidentifier -&gt;false,https://&lt;host&gt;:&lt;port&gt;/nidp/saml2/metadata/spnameidentifier -&gt;true</pre>

The following sample xml file will be displayed when all the SAML tags are set to true and SAML2\_CHANGE\_ISSUER and SAML2\_ATTRIBUTE\_CONSUMING\_INDEX tags are not set.

#### Sample XML File

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" AssertionConsumerServiceIndex="2"
ForceAuthn="false" ID="id5R6u1JFtay7eK.iI97Q3eRI34u8" IssueInstant="2013-01-18T06:11:26Z"
Version="2.0"><saml:Issuer>https://nam.rtresearch.net:8443/nidp/saml2/metadata/saml:Issuer</saml:Issuer></samlp:AuthnRequest>
```

The following sample xml file will be displayed when all the SAML tags are set to false and SAML2\_CHANGE\_ISSUER and SAML2\_ATTRIBUTE\_CONSUMING\_INDEX properties are set in `nidpconfig.properties` file.

#### Sample XML File

```
samlp:AuthnRequest
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"xmlns:saml="urn:oasis:names:tc:SAML:2.0:as
sertion" AssertionConsumerServiceIndex="0"
AttributeConsumingServiceIndex="2"Consent="urn:oasis:names:tc:SAML:2.0:consent:unavailabl
e" ForceAuthn="false" ID="idoeZTKq7FOs5MsCigBBCwp30IqD0"
IsPassive="false"IssueInstant="2013-01-
23T05:25:32Z"ProtocolBindingProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST"Version="2.0"><saml:Issuer>https://saml.mariagerfjord.dk:8443/nidp/saml2/metadata/saml:Issuer</saml:Issuer><samlp:NameIDPolicyAllowCreate="true"Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"SPNameQualifier="https://nam.rtresearch.net:8443/nidp/saml2/metadata/"><samlp:Scoping ProxyCount="5"/></samlp:AuthnRequest>
```

## 7.20 Sample Configurations

- [Section 7.20.1, “Setting Up Google Applications,” on page 259](#)
- [Section 7.20.2, “Setting Up Office 365 Services,” on page 261](#)
- [Section 7.20.3, “Integrating Salesforce With Access Manager By Using SAML 2.0,” on page 261](#)
- [Section 7.20.4, “Integrating Shibboleth Identity Provider With Access Manager,” on page 263](#)

### 7.20.1 Setting Up Google Applications

Google Applications are pre-configured to establish federation with external service providers.

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > [Protocol]**.

For the protocol, click **SAML 2.0**.

General

Local

Liberty

SAML 1.1

SAML 2.0

WS Federation

Brokering

WS-Trust

Trusted Providers

Profiles

New

Delete

Enable

Disable

<input type="checkbox"/>	Name	Enabled	Metadata Expiration Date	Metadata Repository
Identity Providers				
<input type="checkbox"/>	<a href="https://gmc.89-205.labs.blr.novel.com/nidp/saml2/metadata">https://gmc.89-205.labs.blr.novel.com/nidp/saml2/metadata</a>	<input checked="" type="checkbox"/>	Not specified	289CMD

- 2 Click **New > Service Provider**.

### Create Trusted Service Provider

#### Step 1 of 2: Specify name and metadata

Provider Type:

Source:

Name: \*

URL: \*

**NOTE:** By default, the **Provider Type General** is selected.

- 3 Select **Google Application** from the **Provider Type** drop-down list.  
By default, the **Metadata Text** source is selected and the **Text** field is pre-filled with the metadata XML. You should edit the Location in the metadata text and replace YOURDOMAIN with the domain name configured in Google Applications.
- 4 In the **Name** option, specify a name by which you want to refer to the provider and click **Next**.
- 5 Review the metadata certificates and click **Finish**. For Google Applications, the certificates page displayed is empty because the metadata does not contain information about the certificates. The system displays the trusted provider on the protocol page. For example, if you have specified the **Name** as **GoogleApps**, the page displays the trusted service provider when you click **Finish**.

**Figure 7-8** Trusted Service Provider for Google Application/Office 365/Sales Force

Access Manager

Devices

Policies

Auditing

Security

Identity Servers

IDP

General

Local

Liberty

SAML 1.1

SAML 2.0

WS Federation

Brokering

WS-Trust

Trusted Providers

Profiles

New

Delete

Enable

Disable

<input type="checkbox"/>	Name	Enabled	Metadata Expiration Date	Metadata Repository
Identity Providers				
No items				
Service Providers				
<input type="checkbox"/>	<a href="#">GoogleApps</a>	<input checked="" type="checkbox"/>	Not specified	Not Applicable
<input type="checkbox"/>	<a href="#">Office365</a>	<input checked="" type="checkbox"/>	Not specified	Not Applicable
<input type="checkbox"/>	<a href="#">SalesForce</a>	<input checked="" type="checkbox"/>	Not specified	Not Applicable

- 6 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 216](#).

You can configure Access Manager to provide the single sign-on services to Google applications by using Security Assertion Markup Language (SAML) 2.0. For more information, see [Integrating Google Apps and Novell Access Manager using SAML 2.0](#).

## 7.20.2 Setting Up Office 365 Services

Office 365 is pre-configured to establish federation with external service providers. For more information, see [Section 7.20.1, “Setting Up Google Applications,” on page 259](#). In [Step 3 on page 260](#), select **Office 365**. The system displays the trusted provider on the protocol page. For example, if you have specified the **Name** as Office365, the screen displays the trusted service provider **Office365** as in [Figure 7-8 on page 260](#), when you click **Finish**.

Access Manager is compatible with Microsoft Office 365 and provides single sign-on access to Office 365 services.

For more information, see [Chapter 11, “Configuring Single Sign-On for Office 365 Services,” on page 331](#).

## 7.20.3 Integrating Salesforce With Access Manager By Using SAML 2.0

Salesforce.com is pre-configured to establish federation with external service providers.

### Integrating Salesforce With Access Manager By Using SAML 2.0 for Identity Provider Initiated Login

To integrate Salesforce for idpsend, follow the procedure in [Section 7.20.1, “Setting Up Google Applications,” on page 259](#). In [Step 3 on page 260](#), select **Salesforce**. The system displays the trusted provider on the protocol page. For example, if you have specified the **Name** as SalesForce, the screen displays the trusted service provider as in [Figure 7-8 on page 260](#), when you click **Finish**.

Access Manager allows your users to use their existing LDAP credentials for single sign-on access to salesforce.com as well as any Web applications protected by Access Manager.

For information using SAML 2.0 for Identity Provider initiated login, follow the procedure below.

- 1 Create domain in Salesforce.

To enable IDP-initiated login in Salesforce.com, you must enable and configure the **My Domain** option in Salesforce.com. Defining your own domain provides the basis for an IDP-initiated URL.

- 1a Login as administrator. Go to **Administration Setup > Domain Management > My Domain**.

- 1b Specify the subdomain name and check the availability.

- 1c Agree to the terms and conditions and click **Register Domain**.

- 2 If you have already configured your identity provider for Salesforce.com using the wizard, you must update configuration in the identity provider according to the new domain. Perform the following steps.
  - 2a Download the metadata from Salesforce site for your domain. See [Step 3 on page 262](#). Send and import this metadata into your Identity Server Salesforce configuration. For reimporting metadata in Access Manager Identity Server, see [Section 7.9.1, "Viewing and Reimporting a Trusted Provider's Metadata," on page 227](#).
  - 2b Change the Intersite Transfer URL to point to the new domain URL
- 3 Perform [Step 4 on page 263](#) and [Step 5 on page 263](#).
- 4 Update the Identity Server.

## Integrating Salesforce With Access Manager By Using SAML 2.0 for Service Provider Initiated Login

Service provider configuration options offer you more flexibility and control for example, simultaneously federating with more than one Identity Server. Salesforce.com also supports SP-initiated login along with IDP-initiated login. SP-initiated login lets the user use a simple and intuitive URL to access the target application.

Follow the procedure given below to integrate Salesforce with Access Manager by using SAML 2.0 for service provider initiated login. Assume that the user has a Salesforce account.

- 1 Create domain in Salesforce.

To enable SP-initiated login in Salesforce.com, you must enable and configure the **My Domain** option in Salesforce.com. Defining your own domain provides the basis for an SP-initiated URL.

- 1a Login as administrator. Go to **Administration Setup > Domain Management > My Domain**.

- 1b Specify the subdomain name and check the availability.

- 1c Agree to the terms and conditions and click Register Domain.

If you have already configured your identity provider for Salesforce.com using wizard, you must update configuration in the identity provider according to the new domain. Perform the following steps.

---

**NOTE:** Configure SSO configuration. Follow the procedure below to enable SAML support in Salesforce.

1. Go to your Salesforce account and login.
  2. From the left panel, select **Security Control > Single sign setting > Saml Single Sign-on Setting > New** and fill the form.
  3. To enable SAML select **Security Control > Single sign setting > Saml Single Sign-on Setting > Federated Single Sign-On Using SAML > Edit > Enable Saml**.
- 

- 2 Change the Intersite Transfer URL to point to the new domain URL.
- 3 Import Salesforce metadata in Access Manager.

As with any other SAML federation you must configure both your Access Manager Identity Server and Salesforce.com Service Provider (SP) to establish a trust. You now have an option to download your metadata from Salesforce.com. To download your specific metadata go to your Salesforce.com instance.

**3a** Login as administrator. Go to **Administration Setup > Security Controls > Single Sign-On Settings**.

**3b** Select **Name** which you have configured above and **Download Metadata**.

**3c** Reimport this metadata into your service provider configuration in Access Manager assuming that you have created Salesforce using the wizard.

The metadata file you download will include a certificate. For Access Manager to trust or use this certificate, the trusted root certificate chain that minted the certificate must exist in the Access Manager certificate trust stores.

**4** Import certificate in Access Manager, for example, Salesforce.com.

**4a** Open the downloaded metadata .xml file with a file editor and search for the certificate in the X509Certificate element (between <ds:X509Certificate> and </ds:X509Certificate>).

**4b** Copy the information into its own file and give it a .cer file extension. Windows will recognize this as a certificate.

**4c** Double click and open the file.

**4d** Click **Certification Path** to see the chain of authority for the certificate.

You will need the trusted root certificate for every CA in the chain that you see listed.

**4e** In the example above, select the **VeriSign Class 3 International Server CA – G3** and click **View Certificate**.

**4f** Click **Details**.

**5** You can now export the CA trusted root certificate.

**5a** Click **Copy to File...** This will launch the Windows Certificate Export Wizard.

**5b** Select **.DER** encoded when prompted. Give the file a name and save.

**5c** Repeat this process for every CA in the certificate path chain.

**5d** Use the Access Manager Administration Console to import the resulting CA trusted root certificates into your Access Manager keystores.

After importing, add these certificates into the Identity Server Keystore. For more information, see [Managing Certificates and Keystores](#) in the [NetIQ Access Manager 4.0 SP1 Administration Console Guide](#).

Ensure to add Root certificate of Salesforce into your OCSP trust store else, OCSP validation fails and the Identity Server displays an error.

## 7.20.4 Integrating Shibboleth Identity Provider With Access Manager

You can establish a single sign-on exchange between Access Manager SAML 2 service provider and a Shibboleth SAML 2 identity provider.

For more information, see [Integrating Access Manager with Shibboleth's Identity Provider Server](#).

## 7.21 Configuring Multiple SAML 2.0 Service Providers on the Same Host for a Single SAML Identity Provider

When the same Access Manager server hosts more than one SAML service provider and federate with another Access Manager acting as an identity provider for these service providers, Access Manager should send different sets of attributes in SAML 2.0 assertions to these service providers.

Perform the following steps to create multiple service providers on the same Access Manager host:

- 1 To create multiple service providers from the same identity provider metadata, manually modify the identity provider's metadata's entityID for each service provider. You can import the metadata text that was edited into the Access Manager configuration to create service providers with different entity IDs.

For more information about how to create a SAML 2.0 service provider, see [Section 7.3.1, "Creating a Trusted Service Provider for SAML 2.0,"](#) on page 210.

- 2 In the Administration Console of the SAML 2.0 identity provider, click **Devices > Identity Servers > Servers > Edit > SAML 2.0 > Service Provider > Options**.
- 3 Set the SAML2\_AVOID\_AUDIENCE\_RESTRICTION property to true. Setting this property to true avoids audience restriction in the SAML 2.0 assertion.
- 4 To avoid the spnamequalifier attribute in nameidentifier of the assertion, do the following:

- 4a Go to Access Manager Identity Server of the SAML 2.0 service provider and open the `nidpconfig.properties` file.

Linux: `/opt/novell/nids/lib/webapp/WEB-INF/classes/nidpconfig.properties`

Windows: `Novell\Tomcat\webapps\nidp\WEB-INF\classes\nidpconfig.properties`

- 4b Add the following:

`SAML2_AVOID_SPNAMEQUALIFIER_TO = <entityID of the service provider>->true`

For example, if you have configured three service provider and you want to avoid sending assertion to the service provider with entity ID "`https://<service provider host>:<port>/nidp/saml2/metadata/spnameidentifier2`" then add the following entry:

```
SAML2_AVOID_SPNAMEQUALIFIER_TO = https://< service provider host>:<port>/
nidp/saml2/metadata/spnameidentifier1->true,https://< service provider
host>:<port>/nidp/saml2/metadata/spnameidentifier2->true,https://< service
provider host>:<port>/nidp/saml2/metadata/spnameidentifier3->true
```

- 5 Restart the Identity Server.

---

**NOTE:** This is possible only when the identity provider and service providers are deployed on Access Manager.

---



# 8 Configuring WS Federation

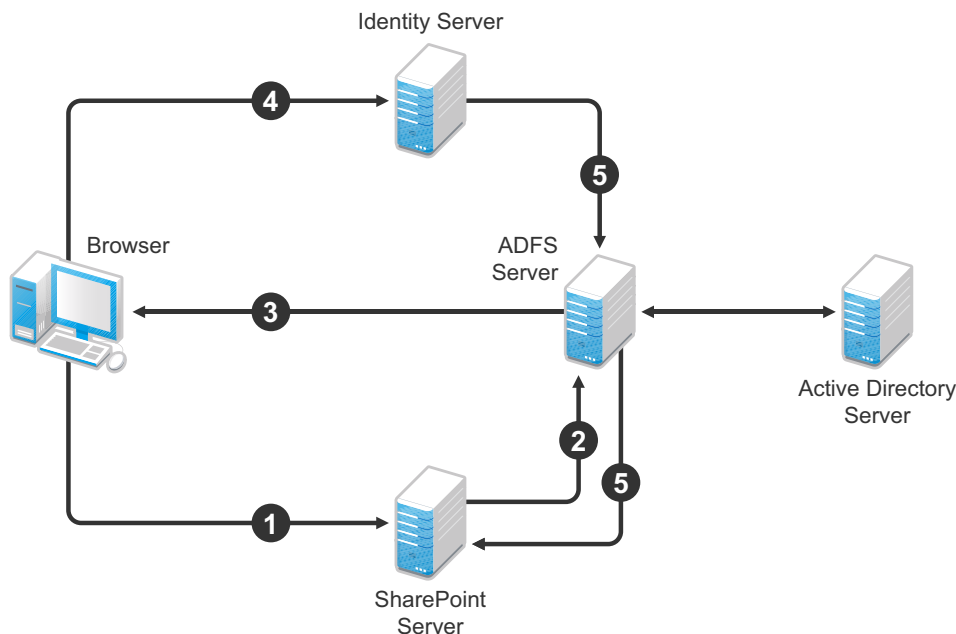
The first two topics in this section describe two different methods for setting up federation with a SharePoint server. The next sections describe how you can manage and modify WS Federation providers and configure Security Token Service (STS). STS is used to process authentication requests received at the Identity Server for the WS Federation protocol.

- [Section 8.1, “Using the Identity Server as an Identity Provider for ADFS,” on page 265](#)
- [Section 8.2, “Using the ADFS Server as an Identity Provider for an Access Manager Protected Resource,” on page 275](#)
- [Section 8.3, “Managing WS Federation Providers,” on page 281](#)
- [Section 8.4, “Modifying a WS Federation Identity Provider,” on page 282](#)
- [Section 8.5, “Modifying a WS Federation Service Provider,” on page 286](#)
- [Section 8.6, “Configuring STS Attribute Sets,” on page 289](#)
- [Section 8.7, “Configuring STS Authentication Methods,” on page 289](#)
- [Section 8.8, “Configuring STS Authentication Request,” on page 290](#)

## 8.1 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 8-1](#) illustrates this configuration.

**Figure 8-1** 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:

- ♦ [Section 8.1.1, “Configuring the Identity Server,” on page 266](#)
- ♦ [Section 8.1.2, “Configuring the ADFS Server,” on page 271](#)
- ♦ [Section 8.1.3, “Logging In,” on page 273](#)
- ♦ [Section 8.1.4, “Troubleshooting,” on page 274](#)

## 8.1.1 Configuring the Identity Server

- ♦ [“Prerequisites” on page 266](#)
- ♦ [“Creating a New Authentication Contract” on page 266](#)
- ♦ [“Setting the WS-Fed Contract to Be the Default Contract” on page 267](#)
- ♦ [“Enabling the WS Federation Protocol” on page 267](#)
- ♦ [“Creating an Attribute Set for WS Federation” on page 268](#)
- ♦ [“Enabling the Attribute Set” on page 268](#)
- ♦ [“Creating a WS Federation Service Provider” on page 269](#)
- ♦ [“Configuring the Name Identifier Format” on page 270](#)
- ♦ [“Setting Up Roles for ClaimApp and TokenApp Claims” on page 270](#)
- ♦ [“Importing the ADFS Signing Certificate into the NIDP-Truststore” on page 270](#)

### 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”](#).
- ♦ You have set up the NetIQ Access Manager 4.0 system with a site configuration that is using SSL in the Identity Server's base URL. See [“Enabling SSL Communication”](#) in the *NetIQ Access Manager 4.0 SP1 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 267](#).

## 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 WS Federation Protocol” on page 267](#).

## Enabling the WS Federation Protocol

Access Manager ships with only SAML 1.1, Liberty, and SAML 2.0 enabled by default. To use the WS Federation protocol, you must enable it on the Identity Server.

- 1 Click the **General** tab.
- 2 In the **Enabled Protocols** section, select WS Federation.
- 3 Click **OK**.
- 4 Update the Identity Server.
- 5 Continue with [“Creating an Attribute Set for WS Federation” on page 268](#).

## Creating an Attribute Set for WS Federation

The WS Federation namespace is `http://schemas.xmlsoap.org/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 namespace:** 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 namespace:** 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 268](#).

## Enabling the Attribute Set

Because the WS Federation protocol uses STS, you must enable the attribute set for STS to use it in an WS Federation relationship.

- 1 On the Identity Servers page, click **Servers > Edit > WS Federation > STS Attribute Sets**.
- 2 Move the WS Federation attribute set to the **Attribute sets** 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

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 8-1** ADFS Resource Server Information

What You Need to Know	Default Value and Description
Provider ID	<b>Default Value:</b> 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 <b>Federation Service URI</b> .
Sign-on URL	<b>Default Value:</b> https://adsresource.treyresearch.net/ads/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 NetIQ 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 NetIQ Identity Server.  This URL is listed in the Properties of the Trust Policy page on the ADFS server. The parameter label is <b>Federation Services endpoint URL</b> .
Logout URL	<b>Default Value:</b> https://adsresource.treyresearch.net/ads/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 <b>Trust Policy</b> on the ADFS Server on the <b>Verification Certificates</b> 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://adsresource.treyresearch.net/ads/ls/`.
  - Logout URL:** (Optional) Specify the URL that the user can use for logging out. The default value is `https://adsresource.treyresearch.net/ads/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 270](#).

## 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 270](#).

## 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 270](#).

## Importing the ADFS Signing Certificate into the NIDP-Truststore

The NetIQ 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 8.1.2, "Configuring the ADFS Server," on page 271](#).

## 8.1.2 Configuring the ADFS Server

The following tasks must be completed on the Trey Research server (adsresouce.treyresearch.net) to establish trust with the NetIQ Identity Server.

- ♦ ["Enabling E-mail as a Claim Type" on page 271](#)
- ♦ ["Creating an Account Partners Configuration" on page 272](#)
- ♦ ["Enabling ClaimApp and TokenApp Claims" on page 272](#)
- ♦ ["Disabling CRL Checking" on page 273](#)

### 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 272](#).

## 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 272](#).

## 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 272](#)).
- 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 273](#).

## 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” \(http://go.microsoft.com/fwlink/?LinkId=68608\)](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 272](#). If the DNS name of your Identity Server is `idp-50.amlab.net`, replace it with `https://idp-50.amlab.net: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
```

### 8.1.3 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 NetIQ 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 274](#) and follow the instructions in [“Common Errors” on page 274](#).

## 8.1.4 Troubleshooting

- ♦ [“Turning On Logging on the ADFS Server” on page 274](#)
- ♦ [“Common Errors” on page 274](#)

### 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 274](#).

### Common Errors

- ♦ [“\[ERROR\] SamlViolatesSaml:” on page 274](#)
- ♦ [“\[ERROR\] Saml contains an unknown NameIdentifierFormat:” on page 274](#)
- ♦ [“CRL Errors” on page 275](#)
- ♦ [“\[ERROR\] EmailClaim.set\\_Email:” on page 275](#)

#### [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 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 273](#).

### [ERROR] EmailClaim.set\_Email:

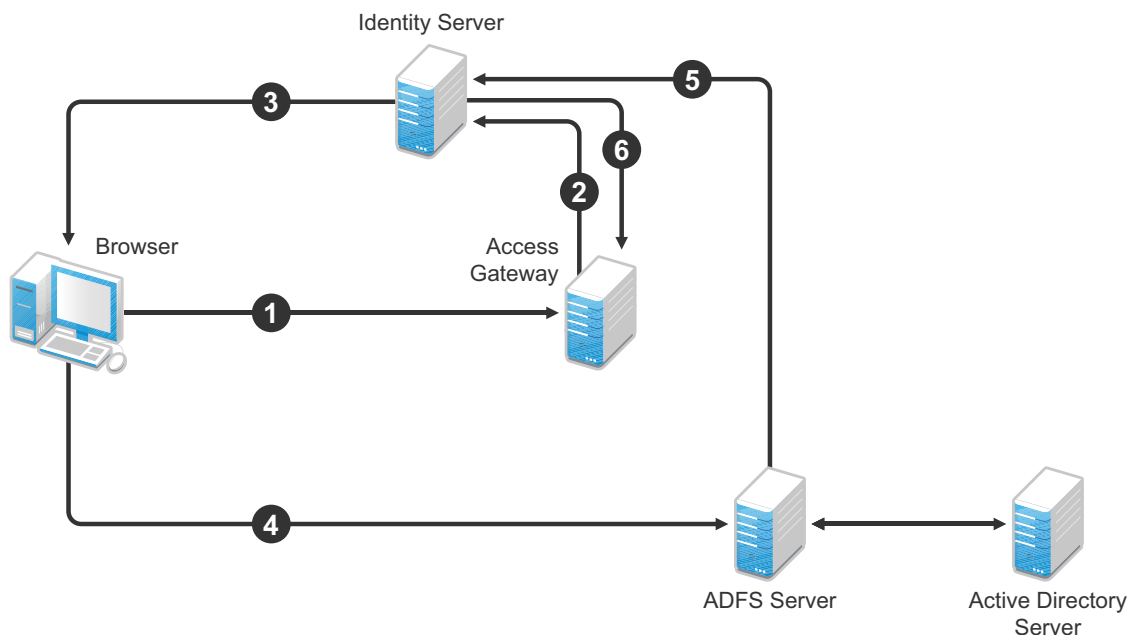
Email 'mPmNXOA8Rv+j16L1iNKn/4HVpfeJ3av1L9c0GQ==' has invalid format

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.

## 8.2 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 8-2** 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 NetIQ 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.

- ♦ [Section 8.2.1, “Configuring the Identity Server as a Service Provider,” on page 276](#)
- ♦ [Section 8.2.2, “Configuring the ADFS Server to Be an Identity Provider,” on page 279](#)
- ♦ [Section 8.2.3, “Logging In,” on page 280](#)
- ♦ [Section 8.2.4, “Additional WS Federation Configuration Options,” on page 280](#)

## 8.2.1 Configuring the Identity Server as a Service Provider

- ♦ [“Prerequisites” on page 276](#)
- ♦ [“Enabling the WS Federation Protocol” on page 276](#)
- ♦ [“Creating a WS Federation Identity Provider” on page 277](#)
- ♦ [“Modifying the User Identification Specification” on page 278](#)
- ♦ [“Importing the ADFS Signing Certificate into the NIDP-Truststore” on page 278](#)

### 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) (<http://go.microsoft.com/fwlink/?linkid=49531>).
- ♦ You have set up the NetIQ Access Manager system with a site configuration that is using SSL in the Identity Server's base URL. See [“Enabling SSL Communication”](#) in the *NetIQ Access Manager 4.0 SP1 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 WS Federation Protocol

Access Manager ships with only SAML 1.1, Liberty, and SAML 2.0 enabled by default. To use the WS Federation protocol, it must be enabled on the Identity Server.

- 1 In the Administration Console, click **Devices > Identity Servers > Edit**.
- 2 In the **Enabled Protocols** section of the General Configuration page, select **WS Federation**.
- 3 Click **OK**.
- 4 Update the Identity Server.
- 5 Continue with [“Creating a WS Federation Identity Provider” on page 277](#).

## Creating a WS Federation Identity Provider

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 8-2** Adatum Values

What You Need to Know	Default Value and Description
Provider ID	<b>Default Value:</b> 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 <b>Properties</b> of the Trust Policy on the ADFS server. The label is <b>Federation Service URI</b> .
Sign-on URL	<b>Default Value:</b> https://adfsaccount.adatum.com/adfs/ls/  The service provider uses this value to redirect the user for login. This URL is listed in the <b>Properties</b> of the Trust Policy on the ADFS server. The label is <b>Federation Services endpoint URL</b> .
Logout URL	<b>Default Value:</b> 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 NetIQ 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 <b>Trust Policy</b> on the ADFS Server on the <b>Verification Certificates</b> 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 278](#).

## Modifying the User Identification Specification

The default settings for user identification are set to do nothing. The user can be 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 278](#).

## 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 8.2.2, “Configuring the ADFS Server to Be an Identity Provider,” on page 279](#).

## 8.2.2 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 279](#)
- ♦ [“Creating a Resource Partner” on page 279](#)

For additional configuration options, see [Section 8.2.4, “Additional WS Federation Configuration Options,” on page 280](#).

### 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 Ensure 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 279](#).

### 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 277](#)). 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 8.2.3, “Logging In,” on page 280](#).

## 8.2.3 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.

## 8.2.4 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 8.4.2, “Configuring the Attributes Obtained at Authentication,” on page 283](#).

For other options that can be modified after you have created the trusted identity server configuration, see [Section 8.4, “Modifying a WS Federation Identity Provider,” on page 282](#).



## 8.3 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 8.3.1, “Creating an Identity Provider for WS Federation,” on page 281](#)
- ♦ [Section 8.3.2, “Creating a Service Provider for WS Federation,” on page 282](#)

**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 8.4, “Modifying a WS Federation Identity Provider,” on page 282](#) or [Section 8.5, “Modifying a WS Federation Service Provider,” on page 286](#).

- 3 Click **OK**, then update the Identity Server.

### 8.3.1 Creating an Identity Provider for WS Federation

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 8.2, “Using the ADFS Server as an Identity Provider for an Access Manager Protected Resource,” on page 275](#).

## 8.3.2 Creating a Service Provider for WS Federation

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 8.1, “Using the Identity Server as an Identity Provider for ADFS,” on page 265](#).

## 8.4 Modifying a WS Federation Identity Provider

This section explains how to modify a WS Federation identity provider after it has been created. [Section 8.3.1, “Creating an Identity Provider for WS Federation,” on page 281](#) explains the steps required to create an identity provider. You can modify the following configuration details:

- ♦ [Section 8.4.1, “Renaming the Trusted Provider,” on page 282](#)
- ♦ [Section 8.4.2, “Configuring the Attributes Obtained at Authentication,” on page 283](#)
- ♦ [Section 8.4.3, “Modifying the User Identification Method,” on page 283](#)
- ♦ [Section 8.4.4, “Viewing the WS Identity Provider Metadata,” on page 284](#)
- ♦ [Section 8.4.5, “Editing the WS Identity Provider Metadata,” on page 285](#)
- ♦ [Section 8.4.6, “Modifying the Authentication Card,” on page 285](#)
- ♦ [Section 8.4.7, “Assertion Validity Window,” on page 286](#)

### 8.4.1 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.

## 8.4.2 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 268](#).
  - 6 Click **OK**, then update the Identity Server.

## 8.4.3 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 about how to create this contract, see [“Creating a New Authentication Contract” on page 266](#).
- 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 13.3, "Defining the User Provisioning Method," on page 375](#).

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 13.1.2, "Configuring the Attribute Matching Method for Liberty or SAML 2.0," on page 372](#).

7 Click **OK** twice, then update the Identity Server.

## 8.4.4 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 8.4.5, “Editing the WS Identity Provider Metadata,” on page 285](#).
- 3 To view information about the signing certificate, click **Certificates**.
- 4 Click **OK** twice.

## 8.4.5 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.

## 8.4.6 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.

## 8.4.7 Assertion Validity Window

You can configure the assertion validity time for WS Federation Provider (SP) to accommodate clock skew between the Service Provider and SAML IDP Server.

To set the assertion validity for WSFed configuration, add the following parameters in the IDP web.xml and restart tomcat:

Add the following parameters in the web.xml below the ldapLoadThreshold context param:

```
<context-param>
  <param-name>wsfedAssertionValidity</param-name>
  <param-value>600</param-value>
</context-param>
```

The value 600 which is configurable denotes seconds.

To restart Tomcat enter the following command:

Linux: `/etc/init.d/novell-idp restart`

Windows:

`net stop Tomcat7`

`net start Tomcat7`

## 8.5 Modifying a WS Federation Service Provider

This section explains how to modify a WS Federation service provider after it has been created. [Section 8.3.2, “Creating a Service Provider for WS Federation,” on page 282](#) explains the steps required to create the service provider. You can modify the following configuration details:

- ♦ [Section 8.5.1, “Renaming the Service Provider,” on page 286](#)
- ♦ [Section 8.5.2, “Configuring the Attributes Sent with Authentication,” on page 287](#)
- ♦ [Section 8.5.3, “Modifying the Authentication Response,” on page 287](#)
- ♦ [Section 8.5.4, “Viewing the WS Service Provider Metadata,” on page 288](#)
- ♦ [Section 8.5.5, “Editing the WS Service Provider Metadata,” on page 288](#)

### 8.5.1 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.

## 8.5.2 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 268](#).
  - 6 Click **OK**, then update the Identity Server.

## 8.5.3 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 8.5.2, “Configuring the Attributes Sent with Authentication,” on page 287](#).

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.

## 8.5.4 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 8.5.5, "Editing the WS Service Provider Metadata," on page 288](#).
- 3 To view information about the signing certificate, click **Certificates**.
- 4 Click **OK** twice.

## 8.5.5 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 **Browse** and follow the prompts.
- 4 To view information about the signing certificate, click **Certificates**.
- 5 Click **OK** twice, then update the Identity Server.

## 8.6 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 and select the attributes and namespace appropriate for the protocol.

- 1 In the Administrations Console, click **Devices > Identity Servers > Edit > WS Federation > STS Attribute Sets**.
- 2 To select a set, move the set from the **Available attribute sets** list to the **Attribute sets** list.

**WS Federation:** There is no default attribute set for WS Federation. For information about how to create the set, see [Section 8.4.2, “Configuring the Attributes Obtained at Authentication,” on page 283](#) and [Section 8.5.2, “Configuring the Attributes Sent with Authentication,” on page 287](#).

- 3 Click **OK**, then update the Identity Server if you have changed the configuration.

## 8.7 Configuring STS Authentication Methods

Use the Authentication Methods page to select the methods that can be used for authentication at the STS. 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 > WS Federation > 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 [NetIQ Access Manager Developer Tools and Examples](#).

- 3 Click **OK**, then update the Identity Server if you have changed the configuration.

## 8.8 Configuring STS 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 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 > WS Federation > 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.

---

# 9 WS-Trust Security Token Service

This section describes WS-Trust Security Token Service (WS-Trust STS) and how to configure it. Topics include:

- [Section 9.1, “Overview,” on page 291](#)
- [Section 9.2, “Benefits,” on page 293](#)
- [Section 9.3, “Scenarios,” on page 293](#)
- [Section 9.4, “Configuring WS-Trust STS,” on page 300](#)
- [Section 9.5, “Configuring Service Providers,” on page 302](#)
- [Section 9.6, “Configuring Web Service Clients,” on page 307](#)
- [Section 9.7, “Renew Token - Sample Request and Response,” on page 309](#)

## 9.1 Overview

Web services are software applications that interact over network by using the Hyper Text Transfer Protocol (HTTP). World Wide Web Consortium (W3C) describes Web services as a standard means of interoperating between software applications running on a variety of platforms and frameworks. A Web service provides a fine-grained service to its client.

Web services use network for communication and data exchange spanning from protected network (intranet) to unprotected network (internet). This demands for security requirements such as client authentication, access control, data integrity, and confidentiality.

You can secure Web services and protect your information against authentication attacks and unauthorized access by using security tokens. A security token contains a set of claims made by a client that includes details such as a user name, password, identity, key, and certificate.

NetIQ Access Manager addresses the need for a secure token mechanism through WS-Trust STS that is based on the WS-Trust protocol. WS-Trust is built on top of the WS-Security specification. It enables Web applications to construct trusted SOAP message exchanges.

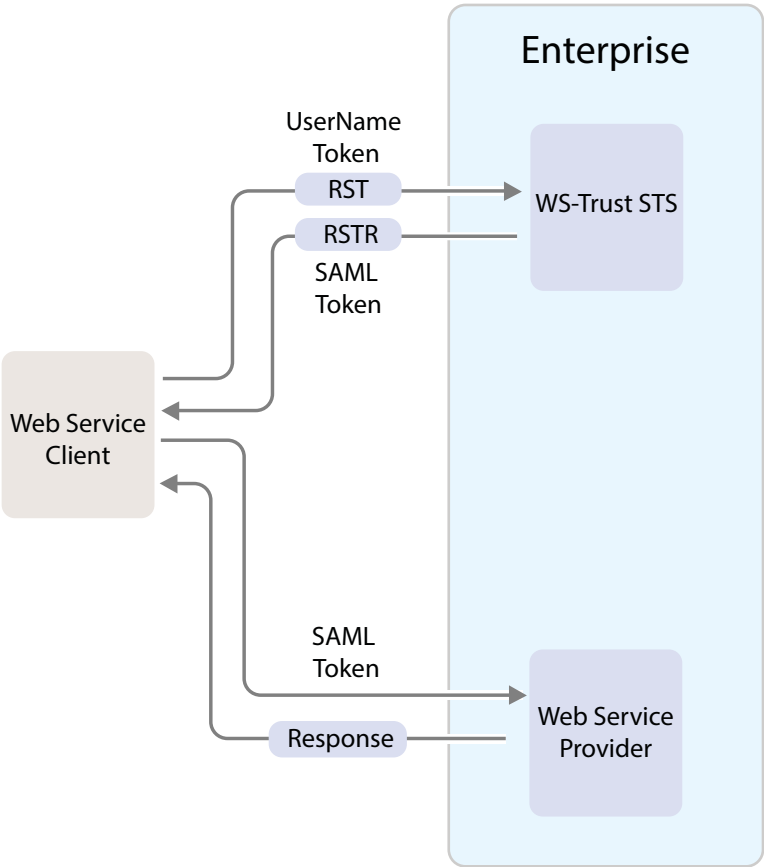
WS-Trust STS provides secure communication and integration between services. It issues and validates security tokens to establish trust relationships between a Web service client and a Web service provider. If the requestor (Web service client) does not have the necessary token to provide required claims to a service, it contacts WS-Trust STS and requests the needed tokens with proper claims. WS-Trust STS may in turn require its own set of claims for authenticating and authorizing the request for security tokens.

### How WS-Trust STS Works

WS-Trust STS allows secure identity propagation and token exchange between Web services. It provides a standard framework for requesting and returning security tokens by using Request Security Token (RST) and Request Security Token Response (RSTR) messages. RST provides the means for requesting a security token from WS-Trust STS. RSTR contains the requested token, claims, and other related information.

The Web service client provides its credentials to WS-Trust STS and gets a SAML token from WS-Trust STS. A trust is established between the Web service provider and WS-Trust STS. The Web service client presents the token from WS-Trust STS to the Web service provider. The Web service provider validates if the token has been issued from WS-Trust STS and grants access to the required service.

The following diagram illustrates an example of how WS-Trust STS facilitates a secure communication between a Web service client and a Web service provider through security tokens:



WS-Trust STS is designed to interoperate with many different Web Service environments and supports SOAP and WS?Trust specifications.

Web services must implement the protocol defined in the WS-Trust 1.3 or 1.4 specification by making assertions based on evidence that it trusts, to whoever trusts it, or to specific recipients. For more information about WS-Trust specification, see [WS-Trust 1.3 Specification](#) and [WS-Trust 1.4 Specification](#).

The following table lists supported SOAP and WS?Trust versions and corresponding namespaces:

Specification	Version	Namespace
Soap	1.1	<a href="http://schemas.xmlsoap.org/soap/envelope/">http://schemas.xmlsoap.org/soap/envelope/</a>
	1.2	<a href="http://www.w3.org/2003/05/soap-envelope">http://www.w3.org/2003/05/soap-envelope</a>
WS-Trust	1.3	<a href="http://docs.oasis-open.org/ws-sx/ws-trust/200512/">http://docs.oasis-open.org/ws-sx/ws-trust/200512/</a>
	1.4	<a href="http://docs.oasis-open.org/ws-sx/ws-trust/200802">http://docs.oasis-open.org/ws-sx/ws-trust/200802</a>

---

**NOTE**

- ♦ WS-Trust STS supports SAML tokens in addition to username tokens.  
WS-Trust STS can issue both SAML 1.1 and SAML 2.0 based tokens.
  - ♦ WS-Trust STS supports issuing, validating and renewing tokens. This release does not support canceling tokens.
  - ♦ Web service clients and Web service providers should be in the same domain. This release does not support multiple domains.
  - ♦ WS-Trust STS does not support password digest-based authentication in username-token authentication policies.
- 

## 9.2 Benefits

WS-Trust STS offers the following benefits:

- ♦ Enables the secure exchange of SOAP messages among Web services.
- ♦ Facilitates identity delegation (through ActAs) and impersonation (through OnBehalfOf) where an authenticated user is granted access to downstream Web services.
- ♦ Reduces complexity for the Web service consumer as the Web service consumer does not need token specific knowledge.

## 9.3 Scenarios

This section describes the basic scenarios supported by WS-Trust STS.

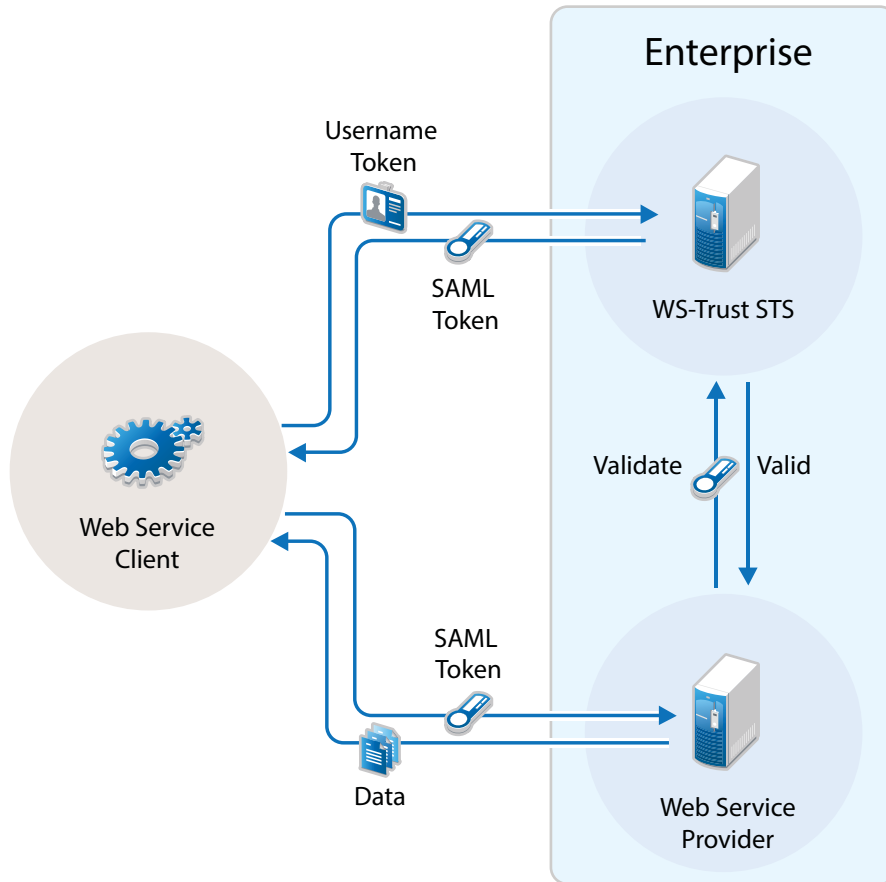
- ♦ [Section 9.3.1, “Scenario 1: Web Service Client Communicating with Token Protected Web Service Provider,” on page 293](#)
- ♦ [Section 9.3.2, “Scenario 2: Web Single Sign-On and STS,” on page 294](#)
- ♦ [Section 9.3.3, “Scenario 3: Identity Delegation and Impersonation,” on page 295](#)
- ♦ [Section 9.3.4, “Renewing a Token,” on page 297](#)
- ♦ [Section 9.3.5, “Authentication Using SAML Tokens,” on page 298](#)

### 9.3.1 Scenario 1: Web Service Client Communicating with Token Protected Web Service Provider

In this scenario, a Web service client situated outside the enterprise tries to access a Web service provider hosted inside the enterprise.

This process consists of requesting a token by means of the request?response message pairs of a Request Security Token (RST) and a Request Security Token Response (RSTR). The tokens are included in SOAP messages.

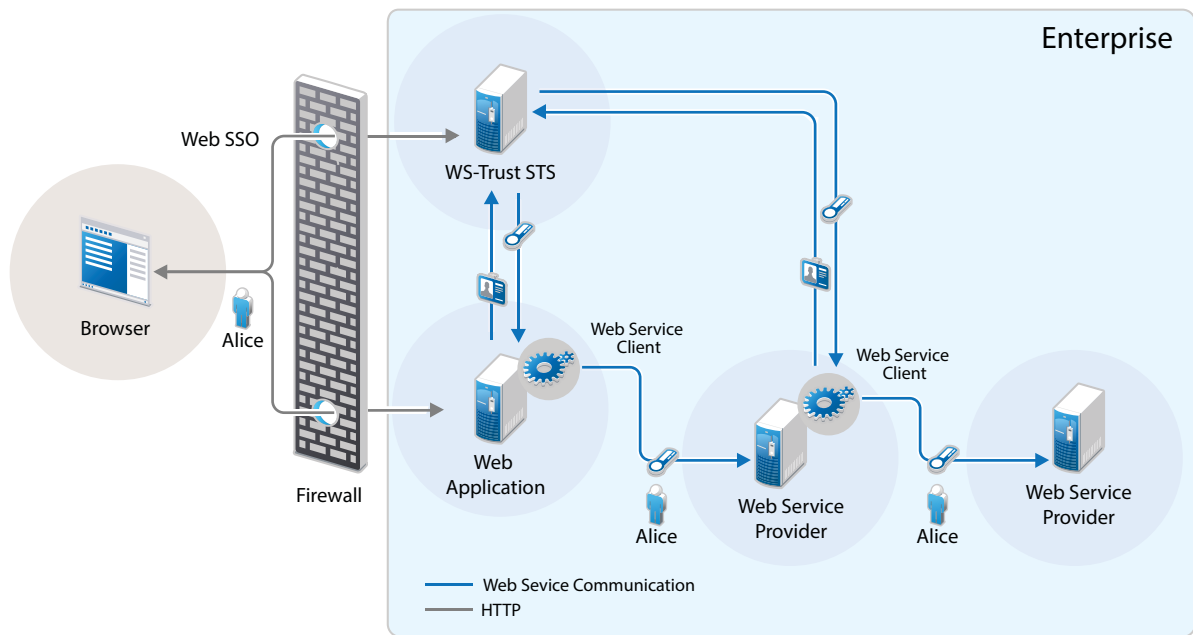
The following diagram illustrates this scenario:



1. A Web service client, which is outside the enterprise, sends its credentials to WS-Trust STS and request for the security token through RST.
2. WS-Trust STS verifies the client's credentials and then issues a security token (SAML token) through RSTR.  
The Web service client caches the security token and then uses it in multiple requests to the Web service provider.
3. The Web service client presents the token to the Web service provider.
4. The Web service provider validates the token and sends the response to the Web service client.

### 9.3.2 Scenario 2: Web Single Sign-On and STS

In this scenario, a Web service client that resides as part of a Web application within an enterprise tries to access services from other Web service providers of the same enterprise. A user named Alice accesses to the Web application through a browser and needs single sign-on access to other applications.



1. A Web application sends a single sign-on authentication request to WS-Trust STS on behalf of Alice.
- The Web application is residing within the enterprise.
2. The Web application passes the credentials corresponding to the single sign-on session to the Web service client.
  3. The Web service client requests for security token by using the passed on credentials.
  4. WS-Trust STS verifies the credentials. After checking the credentials, it verifies if the Web service provider for which the token has been requested for is a trusted service provider. Then it issues a security token consumable by the service provider.
  5. The Web service client residing within the Web application presents the token to the Web service provider. The Web service client caches the security token and then uses it in multiple requests to the Web service provider.
  6. The Web service provider validates the token and sends the response to the Web service client residing within the Web application.

The tokens are included in SOAP messages.

### 9.3.3 Scenario 3: Identity Delegation and Impersonation

In this scenario, a Web service provider requests services from other Web service providers.

The following use-case explains this scenario:

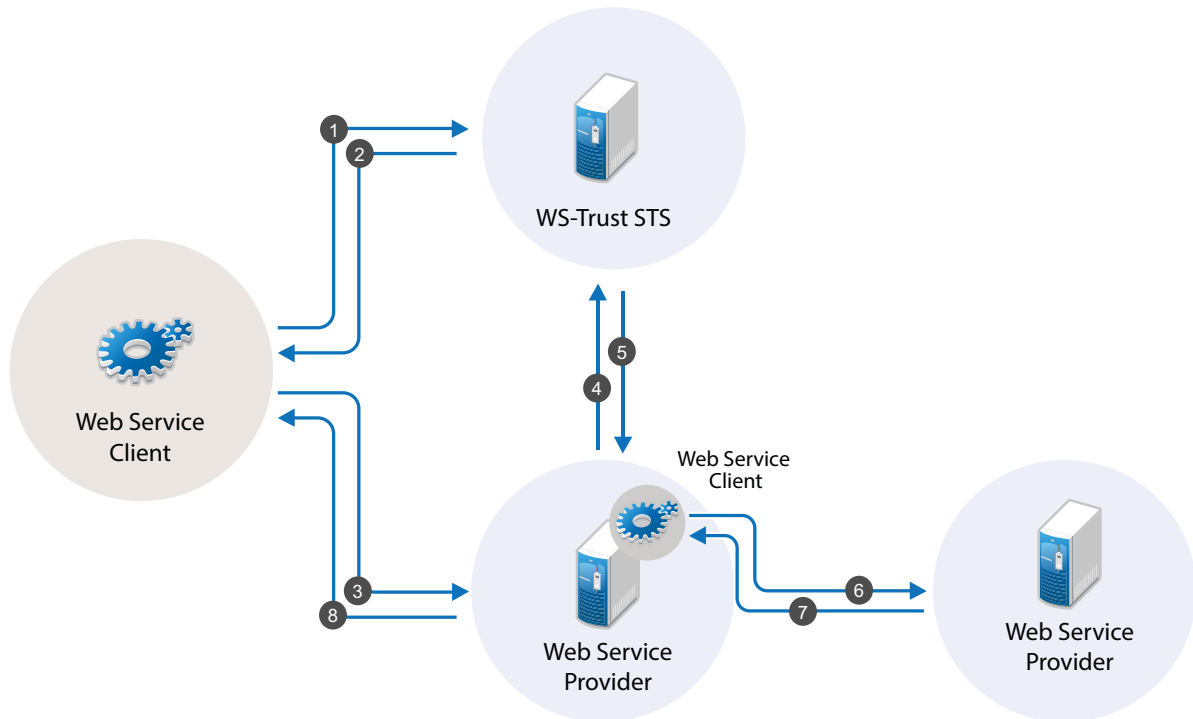
There are two Web service providers called Service1 and Service2 providing some fine-grained service. Both Service1 and Service2 require authentication and trust WS-Trust STS. A Web service client tries to access Service1 and requests for the service. To fulfill the request, Service1 needs to access the service of Service2. Service1 can request a token from WS-Trust STS to access Service2. This exchange of information happens through security tokens embedded in SOAP messages.

This chaining of service request can be any number of nodes based on the implementation.

Requests for tokens can be made in two ways:

- ♦ **By using the ActAs element (Identity Delegation):** The ActAs element is used for delegated requests. Delegated scenarios require composite delegation, where the final recipient of the issued token can see the entire delegation chain. ActAs is commonly used in multi-tiered systems to authenticate and pass information about identities between the tiers without having to pass this information at the application or business logic layer.
- ♦ **By using the OnBehalfOf element (Impersonation):** The OnBehalfOf element is used for proxy requests. OnBehalfOf is used when the identity of only the original client is important. The final recipient of the issued token can only see claims about the original client. The information about intermediaries is not preserved.

The following diagram illustrates the workflow:



The following workflow explains the ActAs scenario:

1. The Web service client sends a RST to WS-Trust STS for its authentication.
2. WS-Trust STS returns a SAML token to the client in the RSTR. Let us refer to this SAML token as token1. The subject of this SAML token is `client`.
3. The client forwards the token1 with its SOAP request to Service1.
4. Then Service1 sends a RST to WS-Trust STS again authenticating itself to the STS. This time the RST contains the token1 inside the ActAs element.
5. WS-Trust STS issues a SAML token (referred to as token2). The subject of this token is `Service1`. It contains an attribute called `ActAs` with the value of `client`.
6. Service1 sends token2 to Service2. By processing token2, Service2 understands that the original requester is `client` and Service1 is acting as the original requester.
7. Service2 sends the response to Service1.
8. Service1 forwards the response to the client.



The following workflow explains the OnBehalfOf scenario:

1. The Web service client sends a RST to WS-Trust STS for its authentication.
2. WS-Trust STS returns a SAML token to the client in the RSTR. Let us refer to this SAML token as token1. The subject of this SAML token is `client`.
3. The client forwards the token1 with its SOAP request to Service1.
4. Then Service1 sends a RST to WS-Trust STS again authenticating itself to the STS. This time the RST contains the token1 inside the OnBehalfOf element.
5. WS-Trust STS issues a SAML token (referred to as token2). The subject of this token is `client`.
6. Service1 sends token2 to Service2. Service2 understands that the original requester is `client` but cannot see the details of Service1.
7. Service2 sends the response to Service1.
8. Service1 forwards the response to the client.

---

**IMPORTANT:** Starting from Access Manager 4.0 SP1 release, the default binding supported is SOAP 1.2. If you want to use SOAP 1.1 instead, perform the following steps on all instances of the Identity Server:

1. Traverse to the `/opt/novell/nam/idp/webapps/nidp/WEB-INF` folder and edit the `sun-jaxws.xml` file.
  2. Remove all instances of bindings from the endpoints in the `sun-jaxws.xml` file and save the changes. A binding is represented by the following line in this file:  

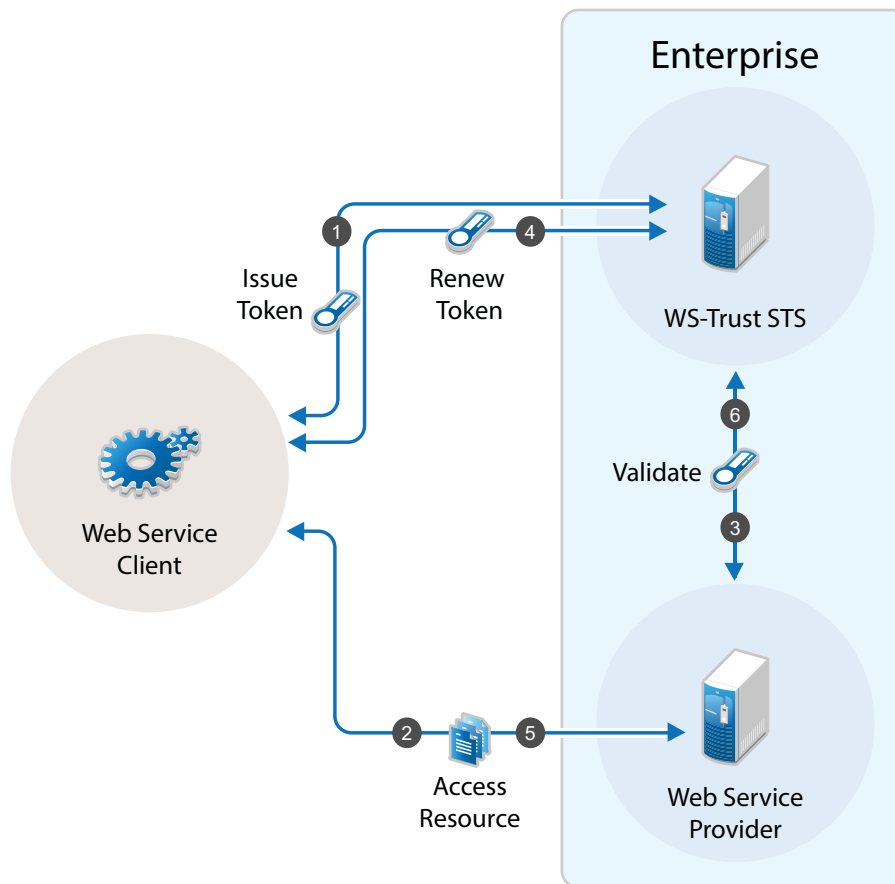
```
binding="http://java.sun.com/xml/ns/jaxws/2003/05/soap/bindings/HTTP/"
```
  3. Restart the Identity Server using the `/etc/init.d/novell-idp restart` command.
- 

### 9.3.4 Renewing a Token

The renew token operation helps in renewing a token issued by WS-Trust Security Token Service(STS). Only a token that is issued by an Identity Server that is part of the same cluster can be renewed. Tokens issued by a different Identity Server in a different cluster or by a third-party STS cannot be renewed.

Each token generated by the STS is valid for the duration specified using the **Token Lifetime** setting. A token can be renewed only before lapse of the expiry period. For example: if the Token Lifetime has been specified as 180 seconds, token renew operation will succeed only till the 179th second.

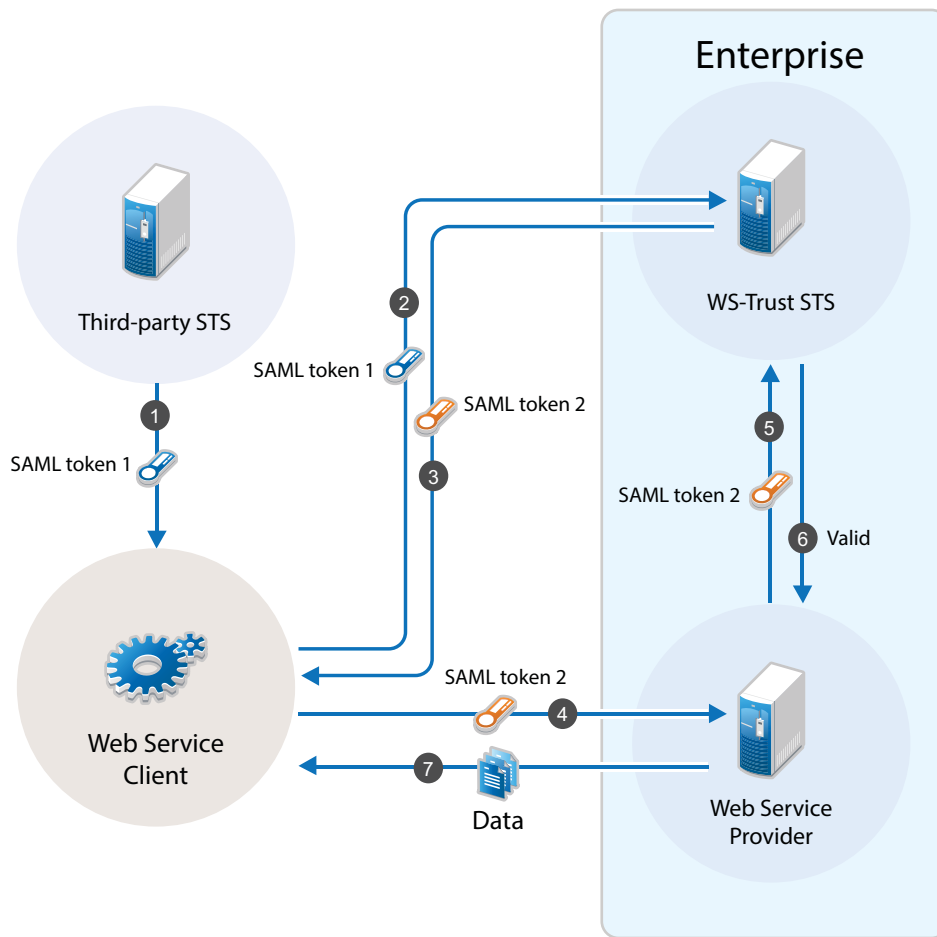
**Workflow:**



1. The Web service client sends a RST to WS-Trust STS for its authentication and WS-Trust STS returns a SAML token to the client in the RSTR
2. Web Service Provider uses the SAML token from STS and requests access to resources hosted on the Web Service provider.
3. The Web Service Provider validates the SAML token and provides access to the resources.
4. When the token is nearing expiry, the Web service client sends a RST to WS-Trust STS to renew the token previously issued. The STS renews the validity of the token and sends a renewed token to the Web Service client for any further requests.
5. Web Service client uses the renewed SAML token from STS and requests access to resources hosted on the Web Service provider.

### 9.3.5 Authentication Using SAML Tokens

WS-Trust STS now accepts SAML tokens issued by a third-party STS for authentication. The tokens can be in SAML 1.1 or SAML 2.0. This is an improvement over the existing feature where only username tokens were accepted for authentication.



#### Workflow:

1. The Web service client sends a RST to third-party STS. The third-party STS returns a SAML token to the client in the RSTR.
2. The Web Service client uses SAML token issued by the third-party STS and requests WS-Trust STS to grant access to resources hosted on the Web Service Provider.
3. WS-Trust STS authenticates the client using the third-party SAML token and issues a new SAML token.
4. The Web Service client uses this new SAML token to get access to resources hosted on the Web Service Provider.
5. The Web Service Provider validates the SAML token with WS-Trust STS.
6. The Web Service client can access the resources on the Web Service Provider if the SAML token is valid.

## 9.4 Configuring WS-Trust STS

Before a Web service can invoke operations on STS, you must enable WS-Trust and configure it in Access Manager. This section discusses the following topics:

- ♦ [Section 9.4.1, “Enabling WS-Trust,” on page 300](#)
- ♦ [Section 9.4.2, “Configuring Access Manager for WS-Trust STS,” on page 300](#)
- ♦ [Section 9.4.3, “Viewing STS Service Details,” on page 301](#)

### 9.4.1 Enabling WS-Trust

Access Manager ships with only SAML 1.1, Liberty, and SAML 2.0 enabled by default. To use the WS-Trust protocol, you must enable it on the Identity Server.

To enable WS-Trust, perform the following steps:

- 1 In the Administration Console, click **Devices > Identity Servers > Edit**.
- 2 In the **Enabled Protocols** section, select **WS-Trust**.
- 3 Click **OK**.
- 4 Update the Identity Server.

### 9.4.2 Configuring Access Manager for WS-Trust STS

To configure WS-Trust STS, perform the following steps:

- 1 In the Administration Console, click **Devices > Identity Servers > Edit**.
- 2 Select **WS-Trust > STS Configuration**.

**General** **Local** **Liberty** **SAML 1.1** **SAML 2.0** **WS Federation** **Brokering** **WS-Trust**

Service Provider Domains | **STS Configuration** | EndPoint Summary

**Policies**

Token Lifetime:  seconds

Token Issuer:

**Authentication Methods**

Authentication Methods

Selected Authentication Methods:

Available Authentication Methods:

- Name/Password - Basic
- Name/Password - Form
- Secure Name/Password - Basic
- Secure Name/Password - Form

**Tokens**

Tokens

Selected tokens:

- SAML1
- SAML2

Available tokens:

OK Cancel Apply

3 Specify the following details:

**Token Lifetime:** Specify the duration in seconds for which the token is valid. The default value is 360 seconds.

**Token Issuer:** Specify the name of the issuer of the authentication token.

**Authentication Methods:** Select methods that can be used for authentication at STS for WS-Trust.

Select and move methods from **Available Authentication Methods** to **Selected Authentication Methods**.

**Tokens:** Select the type of tokens that can be issued for authentication at STS for WS-Trust. SAML 1.1 and SAML 2.0 tokens are supported. If you select both token types, the token type configured in the service provider is returned.

4 Click **Apply**.

## 9.4.3 Viewing STS Service Details

EndPoint URL is the SOAP endpoint of STS. The Web service client and Web service provider must be configured to these endpoints.

In the Administration Console under **Devices > Identity Servers > Edit > WS-Trust > EndPoint Summary**, you can view the following STS EndPoint details:

**Service Name:** The name of the STS service endpoint that is configured in the Web service client.

**Port Name:** The port that STS implements. This is configured in the Web service client.

**MEX EndPoint URI:** The MetadataExchange endpoint of STS.

**WSDL of STS Username authentication:** WSDL location for username authentication. This file is used by applications that use the token service with username authentication.

**WSDL of STS SAML authentication:** WSDL location for SAML. This file is used by applications that use the token service with SAML authentication.

## 9.5 Configuring Service Providers

You require to configure Web service providers to accept tokens issued by an STS. The Web service provider uses an IssuedToken policy for the same. The IssuedToken policy is wrapped in WSDL. For a sample policy, see [Section 9.5.6, “A Sample WS-Policy for Web Service Providers,” on page 306](#).

Configuring a service provider includes adding a service provider domain and then adding a service provider in a configured domain. Access Manager also allows you to modify and delete configured service provider domains and service providers.

- [Section 9.5.1, “Adding a Domain and Assigning WS-Trust Operations,” on page 302](#)
- [Section 9.5.2, “Adding Web Service Providers,” on page 303](#)
- [Section 9.5.3, “Managing Service Provider Domains,” on page 305](#)
- [Section 9.5.4, “Managing Service Providers,” on page 305](#)
- [Section 9.5.5, “Modifying Service Providers,” on page 305](#)
- [Section 9.5.6, “A Sample WS-Policy for Web Service Providers,” on page 306](#)

### 9.5.1 Adding a Domain and Assigning WS-Trust Operations

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > WS-Trust > Service Provider Domain**.
- 2 Click **New > General** to create a general domain. Selecting **New > Office 365** creates an Office 365 domain that can be configured for active authentication. For details on creating an Office 365 domain, see [“Configuring an Office 365 Domain By Using WS-Trust Protocol” on page 332](#)
- 3 Specify the following details:
  - Name:** Specify a name for the domain.
  - WS-Trust Operations:** Select operations in **Available operations** that WS-Trust STS performs for tokens and move these to **Selected operations**.  
The available operations are Issue, Validate, OnBehalfOf, ActAs and [Renew](#).  
If you select OnBehalfOf and Act As the Available operations, additional configuration is required. For more information, see [“Adding Policy for ActAs and OnBehalfOf” on page 304](#)
- 4 Click **Finish**. Continue with creation of a trusted Service Provider. For more information, see [Adding Web Service Providers](#)

## 9.5.2 Adding Web Service Providers

This section discusses how to add service providers for WS-Trust STS. Adding a service provider includes adding service provider EndPoint URL, configuring trust certificates, selecting token types, and customizing attributes.

Perform the following steps:

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > WS-Trust > Service Provider Domain**.
- 2 Select the domain under which you want to configure a service provider.
- 3 Click **Service Provider > New**.
- 4 Specify the following details:
  - Name:** Specify a name for the service provider.
  - Endpoint:** Specify the SOAP endpoint location at the service provider to which SOAP messages are sent.
  - Token Type:** Select the type of token that the service provider will accept or validate.
  - Encrypt Proof Token Using:** Import a certificate from the file system or paste content of the certificate here. This certificate should be configured in the Web service provider and is used for creating the subject confirmation in the SAML token.
- 5 Click **Finish**.
- 6 Select the Service Provider to define the Attributes and Authentication Response. For more information, see [Section 9.5.5, “Modifying Service Providers,” on page 305](#)

## Enabling Delegation and Impersonation

By default, ActAs and OnBehalfOf requests are disabled in the Access Manager Identity Server. To enable delegation and impersonation, you must enable ActAs and OnBehalfOf by performing the following steps:

- 1 Go to **WS-Trust > Service Provider Domain**.
- 2 Click the service provider domain name for which you want to enable ActAs and OnBehalfOf operations.
- 3 Under **WS Trust Operations**, select **ActAs** and **OnBehalfOf** in **Available operations** and move to **Selected operations**.
- 4 Click **OK**.

These operations are restricted to a set of privileged user accounts defined in the policy. You need to configure the allowed user accounts, who can perform ActAs and OnBehalfOf operations, in the `nidconfig.properties` file of each Identity Server installation. For more information, see [“Adding Policy for ActAs and OnBehalfOf” on page 304](#)

## Configuring ActAs to Lookup Multiple User Stores

For ActAs, the username on behalf of whom a client requests for a token must be present in the user store (eDirectory). The default implementation checks for this user only in the default user store. If you want to search the user in a different user store, perform the following steps:

- 1 In the Administration Console, click **Devices > Identity Server > Edit > Local > Classes**.
- 2 Click **New** and specify the following details:

- Display name:** Specify Find\_By\_Username
- Java class:** Select Other
- Java class path:** Specify com.novell.nidp.authentication.local.UserNameAuthenticationClass
- 3 Click **Next > Finish**.
  - 4 Go to **Local > Methods**.
  - 5 Click **New** and select the Find\_By\_Username class.  
For more information about how to configure an authentication method, see [Section 3.3, "Configuring Authentication Methods," on page 125](#).
  - 6 Go to **WS-Trust > STS Configuration**. Move this authentication method in the **Selected Authentication Methods** from **Available Authentication Methods**.

## Adding Policy for ActAs and OnBehalfOf

You must add an policy to allow ActAs and OnBehalfOf operations. The default policy looks for a configuration of allowed user names from the `nidpconfig.properties` file. Allowed usernames are the user accounts that the intermediate Web service provider uses to authenticate with STS when sending a request with ActAs or OnBehalfOf elements. For ActAs and OnBehalfOf, you must specify multiple username values separated with comma. If no value is specified, ActAs and OnBehalfOf are denied.

The `nidpconfig.properties` file is located in the following location:

**Linux:** `/opt/novell/nids/lib/webapp/WEB-INF/classes`.

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

Enable the following attribute by removing the pound (#) symbol from it for allowing ActAs:

```
WSTRUST_AUTHORIZATION_ALLOWED_ACTAS_VALUES=alice,admin
```

Enable the following name-value pair by removing the pound (#) symbol from it for allowing OnBehalfOf:

```
WSTRUST_AUTHORIZATION_ALLOWED_ONBEHALF_VALUES=bob,admin
```

To simplify parameters, you can define only the following parameter:

```
WSTRUST_AUTHORIZATION_ALLOWED_VALUES=alice,admin
```

These users can perform both Actas and onBehalfOf operations.

After editing the file, restart the Identity Server by running the following command:

**Linux:** `/etc/init.d/novell-idp restart`

**Windows:** `net stop Tomcat7`

```
net start Tomcat7
```

After upgrading Access manager, the configuration is set to default values. You must reconfigure the details after each upgrade.



## 9.5.3 Managing Service Provider Domains

The WS-Trust page allows you to create, modify, and delete service provider domains. This page lists all configured service provider domains.

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > WS-Trust > Service Provider Domains**.

The list of all configured service provider domains is displayed.

- 2 Select one of the following actions:
  - ♦ **New:** Select **New > General** to create a general domain. Selecting **New > Office 365** creates a domain that can be configured for single sign-on to Office 365 services. For more on creating Office 365 domain, see [Section 9.5.1, “Adding a Domain and Assigning WS-Trust Operations,” on page 302](#).
  - ♦ **Delete:** Deletes the selected service provider domain.
- 3 Click **OK**, then update the Identity Server.
- 4 Select the Service Provider domain to modify the following details:
  - ♦ **Name:** Modify the name of the service provider domain.
  - ♦ **WS Trust Operations:** Modify the list of selected WS-Trust operations.
- 5 Click **OK**.

## 9.5.4 Managing Service Providers

Access Manager allows you to you to create, modify, and delete trusted service providers. The Service Providers page lists all configured service provider.

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > WS-Trust> Service Provider Domains> [name of the service provider domain] > Service Providers**.

The list of all configured service provider for the selected domain is displayed.

- 2 Select one of the following actions:
  - ♦ **New:** Launches the Create a Service Provider page. For more information, see [Section 9.5.2, “Adding Web Service Providers,” on page 303](#).
  - ♦ **Delete:** Deletes the selected service providers.
- 3 Click **OK**.

## 9.5.5 Modifying Service Providers

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > WS-Trust> Service Provider Domains> [name of the service provider domain] > Service Providers**.

The list of all configured service provider for the selected domain is displayed.

- 2 Click the name of the service provider you want to edit.

### **Configuration > Trust**

You can modify the following details:

- ♦ Name
- ♦ Endpoint
- ♦ Token Type

- ♦ Encrypt Proof Token Using

For more information about these fields, see [Section 9.5.2, “Adding Web Service Providers,” on page 303.](#)

#### Configuration > Attributes

- ♦ Select the Attribute Set and move attributes from the Available list to the **Send with Authentication** pane. This indicates the attributes that you want sent in an assertion to the service provider.

#### Configuration > Authentication Response

- ♦ Specify a 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 [Configuring the Attributes Obtained at Authentication](#)). If you do not select a value for the E-mail, Kerberos, X509, or Unspecified format, a unique value is automatically generated.

---

**IMPORTANT:** In Access Manager 4.0 SP1, the SAML tokens with Name Identifier value other than username do not support ActAs, OnBehalfOf and SAML authentication operations.

---

- 3 Click **Apply**.

## 9.5.6 A Sample WS-Policy for Web Service Providers

You should modify WSDL of a Web service provider to include IssuedTokenPolicy that points to Access Manager WS-Trust STS. To modify WSDL, you require to add a WS-Policy with IssuedTokenElement. The following is a sample configuration:

```
<wsp:Policy wsu:Id="<policy_name>">
  <wsp:ExactlyOne>
    <wsp:All>
      <wsaws:UsingAddressing xmlns:wsaws="http://www.w3.org/2006/05/
addressing/wsd1" wsp:Optional="false"/>
      <sc:KeyStore wspp:visibility="private" alias="xws-security-server"/>
      <sp:SymmetricBinding>
        <wsp:Policy>
          <sp:ProtectionToken>
            <wsp:Policy>
              <sp:IssuedToken sp:IncludeToken="http://
schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
                <sp:RequestSecurityTokenTemplate>
                  <t:TokenType>http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.1#SAMLV1.1</t:TokenType>
                  <t:KeyType>http://schemas.xmlsoap.org/ws/
2005/02/trust/SymmetricKey</t:KeyType>
                  <t:KeySize>256</t:KeySize>
                </sp:RequestSecurityTokenTemplate>
              <wsp:Policy>
                <sp:RequireInternalReference/>
              </wsp:Policy>
            <sp:Issuer>
              <wsaws:Address>https://namtest.com:8443/
nidp/wstrust/sts</wsaws:Address>
            <wsaws:Metadata>
              <wsx:Metadata>
                <wsx:MetadataSection>
                  <wsx:MetadataReference>
                    <wsaws:Address>https://
namtest.com:8443/nidp/wstrust/sts/mex</wsaws:Address>
                  </wsx:MetadataReference>
                </wsx:MetadataSection>
              </wsx:Metadata>
            </wsaws:Metadata>
          </sp:Issuer>
        </wsp:Policy>
      </sp:SymmetricBinding>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

```

        </wsx:MetadataSection>
    </wsx:Metadata>
    </wsaws:Metadata>
    </sp:Issuer>
    </sp:IssuedToken>
    </wsp:Policy>
</sp:ProtectionToken>
<sp:Layout>
    <wsp:Policy>
        <sp:Lax/>
    </wsp:Policy>
</sp:Layout>
<sp:IncludeTimestamp/>
<sp:OnlySignEntireHeadersAndBody/>
<sp:AlgorithmSuite>
    <wsp:Policy>
        <sp:Basic128/>
    </wsp:Policy>
</sp:AlgorithmSuite>
</wsp:Policy>
</sp:SymmetricBinding>
<sp:Wss11>
    <wsp:Policy>
        <sp:MustSupportRefIssuerSerial/>
        <sp:MustSupportRefThumbprint/>
        <sp:MustSupportRefEncryptedKey/>
    </wsp:Policy>
</sp:Wss11>
<sp:Trust10>
    <wsp:Policy>
        <sp:MustSupportIssuedTokens/>
        <sp:RequireClientEntropy/>
        <sp:RequireServerEntropy/>
    </wsp:Policy>
</sp:Trust10>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>

```

## 9.6 Configuring Web Service Clients

Access Manager WS-Trust STS can be accessed from various Web service clients. The following sections provide example configurations and sample code snippets for CXF-based and Metro-based Web service clients:

- [Section 9.6.1, “Configuring Apache CXF-based Web Service Clients,” on page 307](#)
- [Section 9.6.2, “Configuring Metro-based Web Service Clients,” on page 308](#)

### 9.6.1 Configuring Apache CXF-based Web Service Clients

You can configure CXF-based Web service clients either programmatically or through XML configuration files. Below is a sample XML configuration. Add the following features to `cxf.xml` under the top-level beans section:

```

<cxf:bus>
    <cxf:features>
        <cxf:logging />
        <wsa:addressing />
    </cxf:features>
</cxf:bus>

```

Define the STS client with its properties as follows:

```

<jaxws:client name="{<your webservice target namespace>}WebServicePort"
  createdFromAPI="true">
  <jaxws:properties>
  <entry key="ws-security.sts.client">
    <bean class="org.apache.cxf.ws.security.trust.STSClient">
      <constructor-arg ref="cxf" />
      <property name="wsdlLocation"
        value="https://<your idp base url>nidp/wstrust/sts?wsdl" />
      <property name="serviceName" value="{http://www.netiq.com/nam-4-0/
wstrust}SecurityTokenService" />
      <property name="endpointName" value="{http://www.netiq.com/nam-4-0/
wstrust}STS_Port" />

      <property name="wspNamespace" value="http://schemas.xmlsoap.org/ws/2004/
09/policy" />
      <property name="properties">
        <map>
          <entry key="ws-security.username" value="<username to connect to idp>"
/>
          <entry key="ws-security.password" value="<password>" />
          <entry key="ws-security.encryption.properties"
value="clientKeystore.properties" />
          <entry key="ws-security.encryption.username" value="mystskey" />
          <entry key="soap.force.doclit.bare" value="true" />
          <entry key="soap.no.validate.parts" value="true" />
        </map>
      </property>
    </bean>
  </entry>
</jaxws:clien>

```

You can configure `ws-security.callback-handler` to provide username and password programmatically. You can also configure global sts-client in `cxf.xml` that can be used across multiple Web services.

For more information about configuring Apache CXF-based Web service clients, see (<http://cxf.apache.org/docs/ws-trust.html>).

## 9.6.2 Configuring Metro-based Web Service Clients

You can configure Metro-based clients through NetBeans (an integrated development environment).

- 1 Create a Web service client project in NetBeans.
- 2 Right click the project and click **Create Web Service Client** to create a STS client. Point the WSDL to `http://<name of the identity provider server>:<port>nidp/wstrust/sts?wsdl`.
- 3 Configure the username and password to access WS-Trust STS.  
The user configured needs to get authenticated into Access Manager password-based authentication classes. You can also configure the Callback-based configuration in NetBeans to provide username and passwords dynamically.
- 4 When you create a Web service client for your Web service, which is configured for STS-issued tokens, you need to specify the endpoint URL of WS-Trust STS in the Web service client properties. You can specify this in NetBeans by right clicking **Web Service References** > **Web Service** and selecting **Secure Token Service**.

For more information about configuring Metro-based Web service clients, see *To Specify an STS on the Service Side* and *To Specify an STS on the Client Side* in [Configuring A Secure Token Service \(STS\)](#).

## 9.7 Renew Token - Sample Request and Response

- [Section 9.7.1, "Renew Token - Sample Request," on page 309](#)
- [Section 9.7.2, "Renew Token - Sample Response," on page 311](#)

### 9.7.1 Renew Token - Sample Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RST/Renew</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:9cfedcee-
2ebf-47e0-a24a-45281d785136</MessageID>
    <To xmlns="http://www.w3.org/2005/08/addressing">https://
namsb.blr.novell.com:443/nidp/wstrust/sts</To>
    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
      <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
    </ReplyTo>
    <wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://
docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      <wsu:Timestamp wsu:Id="TS-1">
        <wsu:Created>2014-02-10T23:36:42Z</wsu:Created>
        <wsu:Expires>2014-02-10T24:36:42Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id="UsernameToken-2">
        <wsse:Username>admin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordText">novell</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
      <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Renew</
wst:RequestType>
      <wst:TokenType>urn:oasis:names:tc:SAML:2.0:assertion</wst:TokenType>
      <wst:KeyType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/
SymmetricKey</wst:KeyType>
      <wst:Entropy>
        <wst:BinarySecret Type="http://docs.oasis-open.org/ws-sx/ws-trust/
200512/Nonce">200dAaqhrBJqbouiQTf7D2pXtXR036Wi/yswGeoq7iQ=</wst:BinarySecret>
      </wst:Entropy>
      <wst:RenewTarget>
        <saml2:Assertion ID="nsts657b5f4-9bf0-45b7-9875-07eeb6d65196"
IssueInstant="2014-05-26T10:33:50.564Z" Version="2.0" xmlns:ds="http://www.w3.org/
2000/09/xmldsig#" xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xenc="http://www.w3.org/
2001/04/xmlenc#" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ns10="http://
www.w3.org/2000/09/xmldsig#" xmlns:ns13="http://www.w3.org/2001/10/xml-exc-c14n#"
xmlns:ns3="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd"
xmlns:ns5="http://docs.oasis-open.org/ws-sx/ws-trust/200512/" xmlns:ns9="http://
schemas.xmlsoap.org/ws/2006/02/addressingidentity" xmlns:sc="http://docs.oasis-
open.org/ws-sx/ws-secureconversation/200512" xmlns:trust="http://docs.oasis-
open.org/ws-sx/ws-trust/200512" xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wst="http://
schemas.xmlsoap.org/ws/2005/02/trust" xmlns:wsu="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:S="http://www.w3.org/
2003/05/soap-envelope" xmlns:wssell="http://docs.oasis-open.org/wss/oasis-wss-
wssecurity-secext-1.1.xsd">
          <saml2:Issuer>https://namsb.blr.novell.com/nidp/wstrust/sts</
saml2:Issuer>
          <ds:Signature>
            <ds:SignedInfo>
```

```

10/xml-exc-c14n#"/>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/
xmlsig#rsa-sha1"/>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/
xmlsig#enveloped-signature"/>
<ds:Reference URI="#nsts657b5f4-9bf0-45b7-9875-07eeb6d65196">
  <ds:Transforms>
    <ds:Transform Algorithm="http://www.w3.org/2000/09/
xml-exc-c14n#"/>
  </ds:Transforms>
  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/
xmlsig#sha1"/>
  <ds:DigestValue>Z3S4qxx2wRv0k5np2R6ENkIF9pk=</
ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>

<ds:SignatureValue>ZxegeuD7NXfNRPiaYY3v2Nfo9vTx+ceASiAFBDzOfaWGczHBT0eYU+AQM99vdX1
GCBcdWqO9qQR8
2WP7lmzREC6ndg+8g/zJ6UH+Jzsf05hIxCAu7d7fg5qP5/BP++x8vUlpUQ32D8daxx+GIWuZjl0s
8KhdbgLReYSWyX6PV0UbjbnAtDFaBTTJ5lpEqHdK7FGUiISXg679o16BTJSS/V2bBOrX7cZGRGte
PMBGz19qX0rzoENpLJFpJi23+/wAYaqzz0kyRGeyA0De0ugsqw2XRvUPciaYhbqqOraFUfmpypspC
o7Clzwsvn0lhlqVX/lDBwfLokrBeijsG3FN3Hg==</ds:SignatureValue>
<ds:KeyInfo>
  <ds:X509Data>
    <ds:X509Certificate>MIIFBDCCA+ygAwIBAgIkAhwR/
6b9CQnrHMhXuBSYqOCbHugRb+e4U/9jWi9kAgIWCzKcMA0GCSqG
Sib3DQEBBQUAMDExGjAYBgNVBAsTEU9yZ2FuaXphdGlvbmFsIENBMRMwEQYDVQKFApuYw1zY190
cmVlMB4XDTE0MDUyMTE4NTQwMVoXDTE0MDUyMTE4NTQwMVowHzEdMBsGAlUEAxMUMmFtc2IuYmxy
Lm5vdmVsbC5jb20wggEiMA0GCSqGSIb3DQEBQUAA4IBDwAwggEKAoIBAQDRtsdCFBM3ImpIyRaj
OdFFEYbC/ykQUEZFwGp62BAUxoLIOPmDZyxpbqIh+1462GFBYuCvKLOhnelOGV6Ii/cTabaHko7h
T7cfUC3N4kmhnc3IXWgJodRIXmlaUSYDyd79guyVjG0brOWJmXJxvmlao3p8bfzPLnpkEdJ7c8HM
BRqckecaGT8nbpm1KGZFAstrRRTryu2aG670FP3+MHWZmydqLlvrK1NCfe+7DlpOUwA13sSgMslf
6UCI4E50gn6pQ26rctGKrBsFfrX76t6ESZuaqFLWS+YA1lcWS3irtihT0p2GsoxcJzq+IvHosHY+
pvrt4gcJiZJN6P3e6yrrAgMBAAGjggIUMIICEDAdBgNVHQ4EFgQUpskUiviZfQ7yIDLb9sJT+mZH
knghWwYDVR0jBBgwFoAUF7LP7EF6tU2u2qquPNTvLDdV7e8wggHMBgtghkgBhv3AQkEAQSCAbsw
ggG3BAIBAAEB/xMdTm92ZWxsIFNlY3VyaXR5IEF0dHJpYnV0ZSh0bSkWQ2h0dHA6Ly9kZXZlbG9w
ZXIubm92ZWxsLmNvbS9yZXBvc2l0b3J5L2F0dHJpYnV0ZXMvY2VydGF0dHJzX3YxMC5odG0wggFI
oBoBAQAwdCAGAgEBAGFGMAgwBgIBAQIBCGIBaaEaAQEAMAgwBgIBAQIBADAIMAYCAQECAQACACi
BgIBFwEB/6OCAQSGWAIBAgICAP8CAQADDQCAAAAAAAAAAAAAAAAAADQCAAAAAAAAAADAYMBACAQAC
CH/////////AQEAAGQG8N9IMBgwEAIbAAIIf/////////8BAQACBAbw30ihWAIBAgICAP8CAQAD
DQBAAAAAAAAAAAAAAAAAADQBAAAAAAAAAADAYMBACAQACCH/////////AQEAAGQR/6b9MBgwEAIb
AAIIf/////////8BAQACBBH/pv2itjBMAgECAGeAAgIA/wMNAIAAAAAAAAAAAAAAAAAAMJAIAAAAA
AAAAMBIwEAIbAAIIf/////////8BAQAwEjAQAgEAAgh/////////wEBADANBgkqhkiG9w0BAQUF
AAOCAQEAbA0AdHm5pV6cEwSyOoB3aJfaLegMYPlAuTNK9ajhez9PIHPGSQzNxTRbj3eV9P+ueP7j
i8AFVR3Ej4eA7S1i5kPGuSXhwM6VhSiCn+x+HbpmfDwJdu5EvErjTlbbjRU/4wTRCqKe7l0FqKs
rH+BGnuUJw16l2PM+wJ+sajX7ktzP8rk8CF+cTOe8ggFceUJ4igllMMkVbul1RTggRmpcILNFk57
QdmySozjVok1OVQOzIGcAggPBSZeCumNNP8mQIAMvnwWG0cTvDikMkCV1AzCC0WK0dWM53JZD/aa
tHay9w8QWouU5cJo8B+uSm2vN+53PdtMKW0XhJcXtXpKkg==</ds:X509Certificate>
  </ds:X509Data>
</ds:KeyInfo>
</ds:Signature>
<saml2:Subject>
  <saml2:NameID NameQualifier="">admin</saml2:NameID>
  <saml2:SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
  </saml2:Subject>
  <saml2:Conditions NotBefore="2014-05-26T10:33:50.564Z"
NotOnOrAfter="2014-05-26T10:35:50.564Z">
    <saml2:AudienceRestriction>
      <saml2:Audience>http://164.99.184.228:8080/doubleit/services/
doubleit</saml2:Audience>
    </saml2:AudienceRestriction>
  </saml2:Conditions>
  <saml2:Advice/>
  <saml2:AuthnStatement AuthnInstant="2014-05-26T10:33:50.564Z">
    <saml2:AuthnContext>
<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:1.0:am:password</

```

```

saml2:AuthnContextClassRef>
  </saml2:AuthnContext>
</saml2:AuthnStatement>
<saml2:AttributeStatement>
  <saml2:Attribute AttributeName="emailaddress"
AttributeNamespace="http://schemas.xmlsoap.org/ws/2005/05/identity/claims"
Name="emailaddress" NameFormat="http://schemas.xmlsoap.org/ws/2005/05/identity/claims">
    <saml2:AttributeValue>admin@idp.com</saml2:AttributeValue>
  </saml2:Attribute>
</saml2:AttributeStatement>
</saml2:Assertion>
</wst:RenewTarget>
</wst:RequestSecurityToken>
<ns:RequestSecurityToken xmlns:ns="http://docs.oasis-open.org/ws-sx/ws-trust/200512/" />
</soap:Body>
</soap:Envelope>

```

## 9.7.2 Renew Token - Sample Response

```

<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope" xmlns:wssell="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <S:Header>
    <Action S:mustUnderstand="true" xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/RenewFinal</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:f41d7aeb-6f67-4df3-9fe4-e160889b7efb</MessageID>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:9cfedcee-2ebf-47e0-a24a-45281d785136</RelatesTo>
    <To xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To>
    <wsse:Security S:mustUnderstand="true">
      <wsu:Timestamp wsu:Id="_1" xmlns:ns15="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512" xmlns:ns14="http://schemas.xmlsoap.org/soap/envelope/">
        <wsu:Created>2014-05-26T10:35:41Z</wsu:Created>
        <wsu:Expires>2014-05-26T10:40:41Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </S:Header>
  <S:Body>
    <trust:RequestSecurityTokenResponse xmlns:ns10="http://www.w3.org/2000/09/xmldsig#" xmlns:ns13="http://www.w3.org/2001/10/xml-exc-c14n#" xmlns:ns3="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" xmlns:ns5="http://docs.oasis-open.org/ws-sx/ws-trust/200512/" xmlns:ns9="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity" xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512" xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512" xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
      <trust:TokenType>urn:oasis:names:tc:SAML:2.0:assertion</trust:TokenType>
      <trust:RequestedSecurityToken>
        <saml2:Assertion ID="nsta657b5f4-9bf0-45b7-9875-07eeb6d65196"
IssueInstant="2014-05-26T10:35:41.072Z" Version="2.0" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xenc="http://www.w3.org/2001/04/xmenc">
          <saml2:Issuer>https://namsb.blr.novell.com/nidp/wstrust/sts</saml2:Issuer>
          <ds:Signature>
            <ds:SignedInfo>
              <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />

```

```

        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/
xmldsig#rsa-sha1"/>
        <ds:Reference URI="#nsts657b5f4-9bf0-45b7-9875-07eeb6d65196">
            <ds:Transforms>
                <ds:Transform Algorithm="http://www.w3.org/2000/09/
xmldsig#enveloped-signature"/>
                <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/
xmldsig#sha1"/>
            <ds:DigestValue>bX5LSf0HkUpLsMkU/V+x39P+g=</
ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>QLRRXQ4TzTgM9mVa5UF1p7YRqRvLP/
h3pyP0KVzZXcbCfDmtT4b0l4lqfhNoXL+Ym2iu2VlMIC5I
TRSt6D/y6pfs4/nChMrOuk5spMZYLBe+0Pd1GYhfLGzyh/AONZGsoVrHf1/LitMeTp4Mvmk/hTp
8yTWb0r79Ssz5TEbwJ/NkqFXxa9XffheaTySOfNXQYu3tL1rdp7Zaq5BR7mye00huo6gBTshHTXM
fGPYMu/Sy0kapqTBWHUbwT8FzysBEgELZdquhvt1NOFHqkWAbyP5vExjYx106Z7Fu3LnDSq+m
h19S+VLslbBR2XgNofhw/bFVBboYkzZDT6Ipmg==</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>MIIFBDCCA+ygAwIBAgIkAhwR/
6b9CQnrHMhXuBSYqOCbHugRb+e4U/9jWi9kAgIWCzKcMA0GCSqG
SIb3DQEBBQUAMDExGjAYBgNVBAsTEU9yZ2FuaXphdGlvdG9mFsiENBMRMwEQYDVQKFApuYWlZ
cmVlMB4XDTE0MDUyMTE4NTQwMVoXDTI0MDUyMTE4NTQwMVoHZEEdMBsGA1UEAxMUbmFtc2IuYmxy
Lm5vdmVsbC5jb20wggeiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDRTsdCFBM3ImpIyRAj
OdFFEYbC/ykQUEZFwGp62BAUxoLIOPmDZyxpqIh+1462GFByuCVkLohnelOGV6Ii/cTAbahko7h
T7cfUC3N4kmhnc3IXWgjoDRIXMlaUSYDYd79guyVjG0brOWJmXJxvmlao3p8bFzPLnpkEdJ7c8HM
BRqckecaGT8nbpm1KGZFAstrRRTryu2aG670FP3+MHWZmydqLlvrK1NCfe+7DlpOUwA13sSgMslf
6UCI4E50gn6pQ26rctGKrBsFfrX76t6ESZuaqFLWS+YA1lcWS3irtihT0p2GsoxcJzq+IvHosHY+
pvr4gcJiZJN6P3e6yrrAgMBAAGjggIUMIICEDAdBgNVHQ4EFgQUUpSkUiivZfQ7yIDLb9sJT+mZH
kngwHwYDVROjBBgwFoAUFLP7EF6tU2u2qquPNTvLdV7e8wggHMBgtghkgBhv3AQkEAQSCABsw
ggG3BAIBAABE/xMdTm92ZWxsIFNlY3VyaXR5IEF0dHJpYnV0ZSh0bSkWQ2h0dHA6Ly9kZXZlbG9w
ZXIubm92ZWxsLmNvbS9yZXBvc2l0b3J5L2F0dHJpYnV0ZXMvY2VydGF0dHJzX3YxMC5odG0wggei
oBoBAQAwCDAgAgEBAgFGMAgWBgIBAQIBCGIBaaEaAQEAMAgWBgIBAQIBADAIMAYCAQECAQACQCi
BgIBFwEB/6OCAQSGWAIBAgICAP8CAQADDQCAAAAAAAAAAAAAAAAAADQCAAAAAAAAAADAYMBACAQAC
CH/////////AQEAAGQG8N9IMBgwEAIbAAIIf/////////8BAQACBAbw30ihWAIBAgICAP8CAQAD
DQBAAAAAAAAAAAAAAAAADCBAAAAAAAAAAAAAAAAADAYMBACAQACCH/////////AQEAAGQR/6b9MBgwEAIb
AAIIf/////////8BAQACBBH/pv2iTjBMAgECAGAAgIA/wMNAIAAAAAAAAAAAAAAAAAAMJAIAAAAA
AAAAMBIwEAIbAAIIf/////////8BAQAwEjAQAgEAAgh/////////wEBADANBgkqhkiG9w0BAQUF
AAOCAQEAbA0AdHm5pV6cEwSy0oB3aJfaLegMYPLAuTNK9aJhez9PIHPSQzNxTRbj3eV9P+ueP7j
i8AFVR3Ej4eA7S1i5kPGuSXhwM6VhSIScn+x+HbpnFdWJdu5EvErjTibbjRU/4wTRCqKe7loFqKs
rH+BGNuUJwl6l2PM+wJ+saJX7ktzP8rk8CF+cTOe8ggFcEuJ4ig1lMMkVbul1RTggRmpcILNFk57
QdmySozjVok1OVQOzIGcAggPBSZeCumNNP8mQIAmVnWg0cTvDikMkCV1AzCC0WK0dWM53JZD/aa
tHay9w8QWouU5cJo8B+uSm2vN+53PdtMKWOXhJcXtXpKKg==</ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Signature>
    <saml2:Subject>
        <saml2:NameID NameQualifier="">admin</saml2:NameID>
        <saml2:SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
    </saml2:Subject>
    <saml2:Conditions NotBefore="2014-05-26T10:35:41.072Z"
NotOnOrAfter="2014-05-26T10:37:41.072Z">
        <saml2:AudienceRestriction>
            <saml2:Audience>http://164.99.184.228:8080/doubleit/services/
doubleit</saml2:Audience>
        </saml2:AudienceRestriction>
    </saml2:Conditions>
    <saml2:Advice/>
    <saml2:AuthnStatement AuthnInstant="2014-05-26T10:33:50.564Z">
        <saml2:AuthnContext>
            <saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:1.0:am:password</
saml2:AuthnContextClassRef>
        </saml2:AuthnContext>
    </saml2:AuthnStatement>

```



```

        <saml2:AttributeStatement>
          <saml2:Attribute AttributeName="emailaddress"
AttributeNamespace="http://schemas.xmlsoap.org/ws/2005/05/identity/claims"
Name="emailaddress" NameFormat="http://schemas.xmlsoap.org/ws/2005/05/identity/
claims">
            <saml2:AttributeValue xmlns:soap="http://www.w3.org/2003/05/
soap-envelope">admin@idp.com</saml2:AttributeValue>
          </saml2:Attribute>
        </saml2:AttributeStatement>
      </saml2:Assertion>
    </trust:RequestedSecurityToken>
    <trust:Lifetime>
      <wsu:Created>2014-05-26T10:35:41.071Z</wsu:Created>
      <wsu:Expires>2014-05-26T10:37:41.071Z</wsu:Expires>
    </trust:Lifetime>
  </trust:RequestSecurityTokenResponse>
</S:Body>
</S:Envelope>

```



---

# 10 Configuring Active Directory Federation Services with SAML 2.0

This section describes step-by-step instructions for configuring a basic identity federation deployment between Microsoft Active Directory Federation Services 2.0 (AD FS 2.0) and Access Manager by using the Security Assertion Markup Language (SAML) 2.0 protocol, specifically its Web Browser SSO Profile and HTTP POST binding.

You can configure AD FS 2.0 as the claims provider and Access Manager as the relying party, or you can configure Access Manager as the claims provider and AD FS 2.0 as the relying party or service provider.

- ♦ [Section 10.1, “Prerequisites and Requirements,” on page 315](#)
- ♦ [Section 10.2, “Configuring Access Manager as a Claims or Identity Provider and AD FS 2.0 as Relying Party or Service Provider,” on page 317](#)
- ♦ [Section 10.3, “Configuring AD FS 2.0 as the Claims or Identity Provider and Access Manager as the Relying Party or Service Provider,” on page 325](#)
- ♦ [Section 10.4, “AD FS 2.0 Basics,” on page 329](#)
- ♦ [Section 10.5, “Troubleshooting,” on page 329](#)

## 10.1 Prerequisites and Requirements

- ♦ Two servers, one to host AD FS 2.0 and the other to host Access Manager.
- ♦ AD FS 2.0 is deployed.
- ♦ ADFS 2.0 with WIF is deployed.

The test deployment that was created in the AD FS 2.0 Federation with a Windows Identity Foundation (WIF) application (<http://go.microsoft.com/fwlink/?LinkId=193997>) is used as starting point for this deployment. A single Windows Server 2008 R2 instance (fsweb.contoso.com) is used to host both the AD FS 2.0 federation server and a WIF sample application. It presumes the availability of a Contoso.com domain, in which fsweb.contoso.com is a member server. The same computer can act as the domain controller and federation server in the test deployments.

- ♦ ADFS 2.0 with SharePoint 2010 is deployed.

The test deployment that was created in the Configuring SharePoint 2010 AAM applications with AD FS 2.0 (<http://technet.microsoft.com/en-us/library/gg295319.aspx>) is used as starting point for this deployment. A single Windows Server 2008 R2 instance (fsweb.contoso.com) is used to host the AD FS 2.0 federation server and a Windows Server 2008 R2 instance (SP2010) is used to host the SharePoint 2010 application. It presumes the availability of a Contoso.com domain, in which fsweb.contoso.com is a member server. The same computer can act as the domain controller and federation server in the test deployments.

- ♦ Access Manager is deployed.

The Access Manager environment in this deployment is hosted by a fictitious company called nam.example.com. Only the Identity Server component of Access Manager is required for this federation. For more information about installation and deployment of Access Manager, refer to the Access Manager documentation (<http://www.novell.com/documentation/novellaccessmanager31/>).

---

**NOTE:** You can download the evaluation version of Access Manager from Novell's download portal (<http://download.novell.com>).

---

## 10.1.1 Linux Environment

- ♦ Access Manager 3.1 SP4, 3.1 SP5, 3.2.x, or 4.0.
- ♦ SUSE Linux Enterprise Server (SLES) 11 SP1 64-bit or a higher version.

---

**NOTE:** Access Manager supports both Windows and Linux. This section discusses only the Linux environment.

---

## 10.1.2 IP Connectivity

Ensure that the Access Manager (nam.example.com) and AD FS 2.0 (fsweb.contoso.com) systems have IP connectivity between them. The Contoso.com domain controller, if it is running on a separate computer, does not require IP connectivity to the Access Manager system. If the Access Manager firewall is set up, open the ports required for the Identity Server to communicate with the Administration Console.

For more information about these ports, see “Setting Up Firewalls” in the *NetIQ Access Manager 4.0 SP1 Installation Guide*.

For HTTPS communication, the Access Manager Identity Server uses TCP 8443 by default. Your browsers need to access this port when using the HTTP POST Binding. Or, you can change this port to 443 by using iptables. See [Section 1.6, “Translating the Identity Server Configuration Port,”](#) on [page 39](#).

For back-channel communication with cluster members, you need to open two consecutive ports for the cluster, such as 7801 and 7802. The initial port (7801) is configurable. See [Section 1.1.3, “Configuring a Cluster with Multiple Identity Servers,”](#) on [page 23](#).

All federation servers (AD FS and Access Manager) need access to a reliable Network Time Protocol (NTP) time source.

## 10.1.3 Name Resolution

The hosts file on the AD FS 2.0 computer (fsweb.contoso.com) is used to configure name resolution of the partner federation servers and sample applications.

## 10.1.4 Clock Synchronization

Federation events have a short time to live (TTL). To avoid errors based on time-outs, ensure that both computers have their clocks synchronized.

---

**NOTE:** For information about how to synchronize a Windows Server 2008 R2 domain controller to an Internet time server, see article 816042 in the Microsoft Knowledge Base (<http://go.microsoft.com/fwlink/?LinkID=60402>).

On SLES 11 SP1 64-bit or a higher version, use the command `sntp -P no -p pool.ntp.org` to synchronize time with the Internet time server.

---

## 10.2 Configuring Access Manager as a Claims or Identity Provider and AD FS 2.0 as Relying Party or Service Provider

This section explains how to configure a setup in which an Access Manager user gets federated access to the WIF sample application or SharePoint 2010 through AD FS 2.0. This setup uses the SAML 2.0 POST profile.

- [Section 10.2.1, “Configuring Access Manager,” on page 317](#)
- [Section 10.2.2, “Configuring AD FS 2.0,” on page 319](#)
- [Section 10.2.3, “Example Scenario: Access Manager as the Claims Provider and AD FS 2.0 as the Relying Party,” on page 324](#)

### 10.2.1 Configuring Access Manager

- [Using Metadata to Add a New Service Provider Connection](#)
- [Exporting the Identity Provider Metadata to a File](#)

---

**NOTE:** To deploy this identity federation for Access Manager 3.1 SP4 or higher, create a new contract with the “urn:oasis:names:tc:SAML:2.0:ac:classes:Password” URI and with the name password form method. Configure this contract as the default contract.

---

### Using Metadata to Add a New Service Provider Connection

The first step in configuring Access Manager is to use the AD FS metadata to add a service provider for Access Manager.

- [“Getting the AD FS 2.0 Metadata” on page 317](#)
- [“Using the Metadata to Add a New Service Provider Connection” on page 318](#)
- [“Adding an AD FS Server Trusted Certificate” on page 318](#)
- [“Creating an Attribute Set in Access Manager” on page 318](#)
- [“Configuring the Service Provider in Access Manager” on page 319](#)

### Getting the AD FS 2.0 Metadata

- 1 Access the AD FS server metadata URL at `https://<<ADFS (hostname or IP)/FederationMetadata/2007-06/FederationMetadata.xml`.
- 2 Save the AD FS metadata file.
- 3 Open the saved AD FS metadata file in Notepad, WordPad, or in any XML editor.

- 4 Remove the `<RoleDescriptor>` tags from the metadata. For example, remove the following tags:

```
<RoleDescriptor xsi:type="fed:ApplicationServiceType"
protocolSupportEnumeration=http://.....> .....
```

```
<RoleDescriptor xsi:type="fed:SecurityTokenServiceType"
protocolSupportEnumeration=http://.....> </RoleDescriptor>
```

- 5 Save the changes.

## Using the Metadata to Add a New Service Provider Connection

- 1 In the Access Manager Administration Console, click **Devices > Identity Server > Edit > SAML 2.0**.
- 2 Click **New > Add Service Provider**.
- 3 In the **Name** field, specify a name by which you want to refer to the provider.
- 4 Select **Metadata Text** from the **Source** list.
- 5 Paste the copied AD FS metadata that you saved in [Step 5 on page 318](#) into the **Text** field.
- 6 Click **Next > Finish**.
- 7 Update the Identity Server.

## Adding an AD FS Server Trusted Certificate

- 1 Download the certificate authority (CA) certificate from the AD FS server.
- 2 In the Access Manager Administration Console, click **Security > Certificates > Trusted Roots**.
- 3 Click **Import**.
- 4 Specify a name for the certificate and browse for the ADFS certificate.
- 5 Click **OK**.
- 6 Click **Uploaded AD FS CA**.
- 7 Click **Add to Trusted Store** and select **config store**.
- 8 Update the Identity Server.

## Creating an Attribute Set in Access Manager

- 1 In the Access Manager Administration Console, click **Devices > Identity Servers > Shared Settings > Attribute Sets > click New**.
- 2 Provide the attribute set name as **adfs-attributes**.
- 3 Click **Next** with the default selections.
- 4 In the **Create Attribute Set** section, click **New**.
- 5 Select **Idpattribute mail** from the **Local Attribute** list.
- 6 Specify **emailaddress** in the **Remote attribute** field.
- 7 Select **http://schemas.xmlsoap.org/ws/2008/06/identity/claims/** from the **Remote namespace** list.
- 8 Click **OK**.
- 9 Click **New**.
- 10 Select **All Roles** from the **Local Attribute** list.
- 11 Specify roles in the **Remote Attribute** field.

- 12 Select <http://schemas.xmlsoap.org/ws/2008/06/identity/claims/> from the **Remote namespace** list.
- 13 Click **OK**.
- 14 Update the Identity Server.

## Configuring the Service Provider in Access Manager

- 1 In the Access Manager Administration Console, select the ADFS service provider in the **SAML 2.0** tab.
- 2 Click **Authentication Response**.
- 3 Select **Binding to POST**.
- 4 Specify the name identifier format default value and select **unspecified** along with the defaults.
- 5 Click **Attributes**.
- 6 Select **adfs-attributes** from the **Attribute Set** list.
- 7 Select the required attributes to be sent with authentication. For example, the mail and cn attributes.
- 8 Click **OK**.
- 9 Update the Identity Server.

## Exporting the Identity Provider Metadata to a File

Access <https://<<Identity server IP / dns name>>:8443/nidp/saml2/metadata> in a browser and save the page as an XML file, such as `asnam_metadata.xml`. AD FS 2.0 uses this file to automate the setup of the Access Manager Claims Provider instance.

## 10.2.2 Configuring AD FS 2.0

- ♦ [Using Metadata to Add Claims Provider](#)
- ♦ [Editing Claim Rules for the Claims Provider Trust](#)
- ♦ [Editing Claim Rules for the WIF Sample Application](#)
- ♦ [Editing Claim Rules for the SharePoint 2010 Application](#)
- ♦ [Changing the AD FS 2.0 Signature Algorithm](#)
- ♦ [Disabling CRL Checking in the Linux Identity Server](#)

## Using Metadata to Add Claims Provider

You need to use the metadata import capabilities of AD FS 2.0 to create the Example.com claims provider. The metadata includes the public key that is used to validate security tokens signed by Access Manager.

- ♦ [“Using Metadata to Add a Relying Party” on page 319](#)

## Using Metadata to Add a Relying Party

- 1 In AD FS 2.0, in the console tree, right-click the **Claims Provider Trusts** folder, then click **Add Claims Provider Trust** to start the Add Claims Provider Trust Wizard.
- 2 Click **Start**.

- 3 On the Select Data Source page, select **Import data about the claims provider from a file**.
- 4 In the **Federation metadata file location** field, click **Browse**.
- 5 Navigate to the location where you saved `nam_metadata.xml`, click **Open**, then click **Next**.
- 6 On the Specify Display Name page, type `NAM Example`.
- 7 Click **Next > Next > Close**.

## Editing Claim Rules for the Claims Provider Trust

The following claim rule describes how the data from Access Manager is used in the security token that is sent to the WIF sample application or SharePoint 2010.

- 1 In AD FS 2.0, click **Relying Party Trusts**, right click **WIF Sample App**, and then click **Edit Claim Rules**.

or

In the AD FS 2.0 center pane, under **Claims Provider Trusts**, right-click **NAM Example**, then click **Edit Claim Rules**.

- 2 On the **Acceptance Transform Rules** tab, click **Add Rule**.
- 3 On the Select Rule Template page, select the **Pass Through or Filter an Incoming Claim** option.
- 4 Click **Next**.
- 5 On the Configure Claim Rule page, use the following values:

Name	Value
Claim rule name	Name ID Rule
Incoming claim type	Name ID
Incoming name ID format	Unspecified

- 6 Select the **Pass through all claim values** option and click **Finish**.
- 7 Click **Add Rule**.
- 8 On the Select Rule Template page, select the **Pass Through or Filter an Incoming Claim** option.
- 9 Click **Next**.
- 10 On the Configure Claim Rule page, under **Claim rule name**, use the following values:

Name	Value
Claim rule name	Name Rule
Incoming claim type	Name

- 11 Leave the **Pass through all claim values** option selected and click **Finish**.
- 12 To acknowledge the security warning, click **Yes**.
- 13 Click **OK**.
- 14 Click **Add Rule**.



- 15 On the Select Rule Template page, select the **Pass Through or Filter an Incoming Claim** option.
- 16 Click **Next**.
- 17 On the Configure Claim Rule page, in the **Claim rule name** field, use the following values:

Name	Value
Claim rule name	Email Rule
Incoming claim type	E-Mail Address

- 18 Leave the **Pass through all claim values** option selected and click **Finish**.
- 19 To acknowledge the security warning, click **Yes**.
- 20 Click **OK**.

## Editing Claim Rules for the WIF Sample Application

At this point, incoming claims have been received at AD FS 2.0, but rules that describe what to send to the WIF sample application have not yet been created. You need to edit the existing claim rules for the sample application to take into account the new Access Manager external claims provider.

- 1 In AD FS 2.0, click **Relying Party Trusts**.
- 2 Right-click **WIF Sample App**, then click **Edit Claim Rules**.
- 3 On the **Issuance Transform Rules** tab, click **Add Rule**.
- 4 On the Select Rule Template page, click **Pass Through or Filter an Incoming Claim > Next**.
- 5 On the Configure Claim Rule page, enter the following values:

Name	Value
Claim rule name	Pass Name Rule
Incoming claim type	Name

- 6 Leave the **Pass through all claim values** option selected, then click **Finish**.
- 7 On the **Issuance Transform Rules** tab, click **Add Rule**.
- 8 On the Select Rule Template page, click **Pass Through or Filter an Incoming Claim**.
- 9 Click **Next**.
- 10 On the Configure Claim Rule page, enter the following values:

Name	Value
Claim rule name	Pass Name ID Rule
Incoming claim type	Name ID
Incoming Name ID format	Unspecified

- 11 Leave the **Pass through all claim values** option selected, then click **Finish**.
- 12 Click **OK**.

---

**NOTE:** If you changed the rules while federating AD FS 2.0 with the WIF sample application, ensure that you add the Permit All Users issuance rules back to the WIF sample application. See Step 6: – Change Rules in the *AD FS 2.0 Federation with a WIF Application Step-by-Step Guide* (<http://technet.microsoft.com/en-us/library/adfs2-federation-wif-application-step-by-step-guide%28WS.10%29.aspx>).

Or, as an alternative, add a new Permit or Deny Users Based on an Incoming Claim rule allowing incoming Name ID = john@example.com to access the application.

---

## Editing Claim Rules for the SharePoint 2010 Application

At this point, incoming claims have been received at AD FS 2.0, but the rules that describe what to be sent to the SharePoint 2010 application have not yet been created. You need to edit the existing claim rules for the SharePoint 2010 application, which is added as relying party to ADFS 2.0, to configure the new Access Manager external claims provider.

- ♦ “Editing the Claim Rules for the SharePoint 2010 Application” on page 322

### Editing the Claim Rules for the SharePoint 2010 Application

- 1 In AD FS 2.0, click **Relying Party Trusts**.
- 2 Right-click **SP2010**, then click **Edit Claim Rules**.
- 3 On the **Issuance Transform Rules** tab, click **Add Rule**.
- 4 On the Select Rule Template page, click **Pass Through or Filter an Incoming Claim > Next**.
- 5 On the Configure Claim Rule page, enter the following values:

Name	Value
Claim rule name	Pass eMail Rule
Incoming claim type	Email Address

- 6 Leave the **Pass through all claim values** option selected and click **Finish**.

## Changing the AD FS 2.0 Signature Algorithm

By default, Access Manager uses the Secure Hash Algorithm 1 (SHA-1) for signing operations. By default, AD FS 2.0 expects partners to use SHA-256. Complete the following steps to set AD FS 2.0 to expect SHA-1 for interoperability with Access Manager.

---

**NOTE:** The same procedure is recommended for AD FS 2.0 relying party trusts that use Access Manager. If the Access Manager SP signs authnRequests, artifact resolution requests, or logout requests, AD FS 2.0 errors occur unless this signature algorithm setting is changed.

---

- 1 In AD FS 2.0, click **Claims Provider Trusts**.
- 2 Right-click **NAM Example > Properties**.
- 3 On the **Advanced** tab, select **SHA-1** in the **Secure Hash Algorithm** list.
- 4 Click **OK**.

## Using Certificates and Certificate Revocation Lists

For security reasons, production federation deployments require the use of digitally signed security tokens, and optionally allows encryption of the security token contents. Self-signed private key certificates, which are generated from inside the AD FS 2.0 and Access Manager products, are used for signing security tokens.

As an alternative, organizations can use a private key certificate that is issued by a certificate authority (CA) for signing and encryption. The primary benefit of using CA-issued certificates is the ability to check for possible certificate revocation against the certificate revocation list (CRL) from the issuing CA. Also, to avoid the untrusted certificate messages in browsers, the trusted root certificate of the CA must also be imported into your browsers. Many well-known CA's trusted roots are included with common browsers. Using one of these existing CAs to mint your certificates also prevents the untrusted certificate messages.

In AD FS 2.0 and in Access Manager, CRL checking is enabled by default for all partner connections, if the certificate being used by the partner includes a CRL Distribution Point (CDP) extension. This has implications in federation deployments between Access Manager and AD FS 2.0:

- ♦ If a signing/encryption certificate provided by one side of a federation includes a CDP extension, that location must be accessible by the other side's federation server. Otherwise, CRL checking fails, resulting in a failed access attempt. The CDP extensions are added by default to certificates that are issued by Active Directory Certificate Services (AD CS) in Windows Server 2008 R2.
- ♦ If the signing/encryption certificate does not include a CDP extension, no CRL checking is performed by AD FS 2.0 or Access Manager.

### Disabling CRL Checking in the Linux Identity Server

- 1 In Access Manager 3.1 SP4 or 3.1 SP5, modify the `/opt/novell/nam/idp/conf/tomcat7.conf` file and add `JAVA_OPTS="{JAVA_OPTS} -Dcom.novell.nidp.serverOCSPCRL=false"`  
or  
In Access Manager 3.2.x or 4.0, modify `/opt/novell/nam/idp/conf/tomcat7.conf` and add `JAVA_OPTS="{JAVA_OPTS} -Dcom.novell.nidp.serverOCSPCRL=false"`
- 2 To apply the changes, restart the Identity Server by running the `/etc/init.d/novell-idp restart` command.

### Disabling CRL Checking in AD FS 2.0

- 1 Click **Start > Administrative Tools > Windows PowerShell Modules**.
- 2 Enter the following command at the PowerShell command prompt:  

```
set-ADFSClaimsProviderTrust -TargetName "NAM Example"  
-SigningCertificateRevocationCheck None
```

---

**NOTE:** You can make many configuration changes to AD FS 2.0 by using the Windows PowerShell command line and scripting environment. For more information, see the AD FS 2.0 Windows PowerShell Administration section of the *AD FS 2.0 Operations Guide* (<http://go.microsoft.com/fwlink/?LinkId=194005>) and the AD FS 2.0 Cmdlets Reference (<http://go.microsoft.com/fwlink/?LinkId=177389>).

---

## 10.2.3 Example Scenario: Access Manager as the Claims Provider and AD FS 2.0 as the Relying Party

- ♦ [“Accessing the WIF Sample Application” on page 324](#)
- ♦ [“Accessing the SharePoint 2010 Application” on page 324](#)

### Accessing the WIF Sample Application

In this scenario, John from Example.com accesses the Contoso WIF sample application.

---

**NOTE:** Clear all the cookies in the Internet Explorer on the AD FS 2.0 computer (fsweb.contoso.com). To clear the cookies, click **Tools > Internet Options > Delete** under **Browsing History**, and then select cookies for deletion.

---

- 1 On the AD FS 2.0 computer, open a browser window, then navigate to <https://fsweb.contoso.com/ClaimsAwareWebAppWithManagedSTS/default.aspx>.  
The first page prompts you to select your organization from a list.
- 2 Select **NAM Example**, then click **Continue** to sign in.  
When only one Identity Provider is available, AD FS 2.0 forwards the request to that Identity Provider by default.
- 3 The NAM login page appears. Type the user name john, type the password test, then click **Login**.

### Accessing the SharePoint 2010 Application

The user's email ID is used as the mapped attribute to access the SharePoint 2010 application. Assume that a user is created in the NetIQ Identity Server. The email ID configured for this user is namuser1@namidp.com.

---

**NOTE:** Clear all the cookies in the Internet Explorer on the AD FS 2.0 computer (fsweb.contoso.com). To clear the cookies, click **Tools > Internet Options > Delete** under **Browsing History**, then select cookies for deletion.

---

- 1 Ensure that an email ID has been configured for the user in the Access Manager user store.  
For this example, use namuser1@namidp.com.
- 2 Access the SharePoint 2010 application.  
The user is redirected to AD FS 2.0.
- 3 Select **NetIQ Identity Server**.  
The user is redirected to the NAM IDP nidp page for authentication.
- 4 Provide namuser1 as the username and password.  
After authentication, the user is redirected to the SharePoint application.

## 10.3 Configuring AD FS 2.0 as the Claims or Identity Provider and Access Manager as the Relying Party or Service Provider

This section explains how to configure an application through AD FS 2.0 that gets federated access to an application by using Access Manager. The setup uses the SAML 2.0 POST profile.

- ♦ [Section 10.3.1, “Configuring Access Manager,” on page 325](#)
- ♦ [Section 10.3.2, “Configuring AD FS 2.0,” on page 326](#)

### 10.3.1 Configuring Access Manager

The AD FS metadata is used to add an Identity Provider to Access Manager.

- ♦ [“Getting the AD FS 2.0 Metadata” on page 325](#)
- ♦ [“Using the Metadata to Add a New Identity Provider Connection” on page 325](#)
- ♦ [“Adding the AD FS Server Trusted Certificate” on page 326](#)
- ♦ [“Configuring the Identity Provider in Access Manager” on page 326](#)

#### Getting the AD FS 2.0 Metadata

- 1 Access the AD FS server metadata by going to `https://<ADFS hostname or IP>/FederationMetadata/2007-06/FederationMetadata.xml`
- 2 Save the AD FS metadata data.
- 3 Open the AD FS metadata file in Notepad, WordPad, or an XML editor).
- 4 Remove the `<RoleDescriptor>` tags from the metadata.

For example, remove the following tags:

```
"<RoleDescriptor xsi:type="fed:ApplicationServiceType"
    protocolSupportEnumeration=http://.....> .....</
RoleDescriptor>

"<RoleDescriptor xsi:type="fed:SecurityTokenServiceType"
    protocolSupportEnumeration=http://.....> </RoleDescriptor>
```

- 5 Save the changes.

#### Using the Metadata to Add a New Identity Provider Connection

- 1 In the Access Manager Administration Console, select **Devices > Identity Server**.
- 2 Click **Edit**.
- 3 Select **SAML 2.0**.
- 4 Click **New > Identity Provider**.
- 5 Specify the name as **ADFS** in the **Name** field.
- 6 Select **Metadata Text** from the **Source** list.
- 7 Paste the copied ADFS metadata that you saved in [Step 5 on page 325](#) into the **Text** field.
- 8 Click **Next**.
- 9 Specify an alphanumeric value that identifies the card in the **ID** field.

- 10 Specify the image to be displayed on the card in the **Image** field.
- 11 Update the Identity Server.

## Adding the AD FS Server Trusted Certificate

- 1 Retrieve the AD FS server's CA trusted root certificate.
- 2 In the Access Manager Administration Console, select **Security > Certificates**.
- 3 Select **Trusted Roots**.
- 4 Click **Import**.
- 5 Specify the certificate name, and browse for the AD FS certificate authority.
- 6 Click **OK**.
- 7 Click **uploaded AD FS CA**.
- 8 Click **Add to Trusted Store and select config store**.
- 9 Update the Identity Server.

## Configuring the Identity Provider in Access Manager

- 1 Select the **AD FS Identity Provider** in the **SAML 2.0** tab.
- 2 Click **Authentication Card > Authentication Request**.
- 3 Select **Response Protocol Binding to POST**.
- 4 Select **NAME Identifier Format as Transient**.
- 5 Click **OK**.
- 6 Update the Identity Server.

### 10.3.2 Configuring AD FS 2.0

- ♦ [“Using the Metadata to Add a Relying Party” on page 326](#)
- ♦ [“Editing Claim Rules for a Relying Party Trust” on page 327](#)
- ♦ [“Disabling the Certificate Revocation List” on page 328](#)
- ♦ [“AD FS 2.0 Encryption Strength” on page 328](#)

## Using the Metadata to Add a Relying Party

The metadata import capability of AD FS 2.0 is used to create a relying party. The metadata includes the public key that is used to validate security tokens signed by Access Manager.

- 1 In AD FS 2.0, right-click the **Relying Party Trusts** folder, then click **Add Relying Party Trust** to start the Add Relying Party Trust Wizard.
- 2 Click **Start**.
- 3 On the Select Data Source page, select **Import data about the claims provider from a file**.
- 4 In the **Federation metadata file location** section, click **Browse**.
- 5 Navigate to the location where you saved `nam_metadata.xml` earlier, select the file, then click **Open > Next**.

- 6 On the Specify Display Name page, specify NAM Example.
- 7 Click **Next** > **Next** > **Close**.

## Editing Claim Rules for a Relying Party Trust

The data from AD FS is used in the security token that is sent to Access Manager.

- ♦ [“Editing the Claim Rule for a Relying Party Trust” on page 327](#)
- ♦ [“Changing the AD FS 2.0 Signature Algorithm” on page 328](#)

### Editing the Claim Rule for a Relying Party Trust

- 1 The Edit Claim Rules dialog box should already be open. If not, in the AD FS 2.0 center pane, under **Relying Party Trusts**, right-click **NAM Example**, then click **Edit Claim Rules**.
- 2 On the **Issuance Transform Rules** tab, click **Add Rule**.
- 3 On the Select Rule Template page, leave the **Send LDAP Attributes as Claims** option selected, then click **Next**.
- 4 On the Configure Claim Rule page, specify `Get attributes` in the **Claim rule name** field.
- 5 Select **Active Directory** from the **Attribute Store** list.
- 6 In the **Mapping of LDAP attributes** section, create the following mappings:

LDAP Attribute	Outgoing Claim Type
User-Principal-Name	UPN
E-Mail-Address	E-Mail Address

- 7 Click **OK**.
- 8 Click **Apply** > **OK**.
- 9 On the **Issuance Transform Rules** tab, click **Add Rules**.
- 10 On the Select Rule Template page, select **Transform an Incoming Claim**, then click **Next**.
- 11 On the Configure Claim Rule page, use the following values:

Name	Value
Claim rule name	Mapping To Transient Name Identifier
Incoming Claim Type	UPN
Outgoing Claim Type	Name ID
Outgoing name ID format	Transient Identifier

- 12 Select **Pass Through All Claims**, then click **OK**.
- 13 Click **Apply** > **OK**.

## Changing the AD FS 2.0 Signature Algorithm

By default, Access Manager uses the Secure Hash Algorithm 1 (SHA-1) for signing operations. By default, AD FS 2.0 expects partners to use SHA-256.

Perform the following steps to setup AD FS 2.0 to expect SHA-1 for interoperability with the Access Manager Identity Provider:

- 1 In AD FS 2.0, click **Claims Provider Trusts** > right-click **Ping Example** > **Properties**.
- 2 On the **Advanced** tab, select **SHA-1** in the **Secure Hash Algorithm** list.
- 3 Click **OK**.

## Disabling the Certificate Revocation List

- ♦ [“Disabling the CRL Checking Option in the Linux Identity Provider” on page 328](#)
- ♦ [“Disabling the CRL Checking Option in AD FS 2.0” on page 328](#)

For more information about signing and encryption certificates, see [“Using Certificates and Certificate Revocation Lists” on page 323](#).

## Disabling the CRL Checking Option in the Linux Identity Provider

In Access Manager 3.1 SP4 or 3.1 SP5, modify the `/opt/novell/nam/idp/conf/tomcat7.conf` file and add `JAVA_OPTS="{JAVA_OPTS} -Dcom.novell.nidp.serverOCSPCRL=false"`

In Access Manager 3.2.x or 4.0, modify `/opt/novell/nam/idp/conf/tomcat7.conf` and add `JAVA_OPTS="{JAVA_OPTS} -Dcom.novell.nidp.serverOCSPCRL=false"`

## Disabling the CRL Checking Option in AD FS 2.0

- 1 Click **Start** > **Administrative Tools** > **Windows PowerShell Modules**.
- 2 Enter the following command at the PowerShell command prompt:

```
set-ADFSRelyingPartyTrust -TargetName "NAM Example"  
-SigningCertificateRevocationCheck None
```

## AD FS 2.0 Encryption Strength

In AD FS 2.0, encryption of the outbound assertions is enabled by default. Assertion encryption occurs for any relying party or service provider for which AD FS 2.0 possesses an encryption certificate.

AD FS 2.0 uses 256-bit Advanced Encryption Standard (AES) keys or AES-256 for encryption. In contrast, Failing to reconcile these conflicting defaults can result in the failed SSO attempts. To resolve this issue, disable the encryption in AD FS 2.0.

- 1 In AD FS 2.0, click **Start** > **Administrative Tools** > **Windows PowerShell Modules**.
- 2 Enter the following command in at the PowerShell command prompt:

```
set-ADFSRelyingPartyTrust -TargetName "NAM Example"  
-EncryptClaims $False
```



## 10.4 AD FS 2.0 Basics

- ♦ [Section 10.4.1, “Configuring the Token-Decrypting Certificate,” on page 329](#)
- ♦ [Section 10.4.2, “Adding CA Certificates to AD FS 2.0,” on page 329](#)
- ♦ [Section 10.4.3, “Debugging AD FS 2.0,” on page 329](#)

### 10.4.1 Configuring the Token-Decrypting Certificate

1 Open the AD FS 2.0 Management tool, then click **Start > Administrative Tools > AD FS 2.0 Management**.

2 In the left pane, expand the **Service** folder and click **Certificates**.

3 In the **Certificates** section, select **Add Token-Decrypting Certificate**.

4 (Conditional) If you see an error prompting you to run certain commands during the token-decrypting process, run the following PowerShell commands:

```
Add-PSSnapin Microsoft.Adfs.PowerShell
```

```
Set-ADFSProperties -AutoCertificateRollover $false
```

These commands allow you to select other certificates. The certificate must be installed on the server. The certificates are configured on the IIS Manager.

5 Click **Start > Administrative Tools > Internet Information Services (IIS) Manager**.

6 Click **ServerName**.

7 Click **Server Certificates** in the IIS section.

### 10.4.2 Adding CA Certificates to AD FS 2.0

1 In Windows, **Start > Run > mmc**.

2 Attach snapshot certificates as service.

3 Select **AD FS**.

4 Import the CA certificate to trusted authorities.

### 10.4.3 Debugging AD FS 2.0

1 In the **Event Viewer**, click **Applications > AD FS**. You can access the troubleshooting help at “Troubleshooting certificate problems with AD FS 2.0” (<http://technet.microsoft.com/en-us/library/adfs2-troubleshooting-certificate-problems%28WS.10%29.aspx>).

Power Shell Commands Help:

- ♦ *Using Windows PowerShell for AD FS2.0* (<http://technet.microsoft.com/en-us/library/adfs2-help-using-windows-powershell%28WS.10%29.aspx>)
- ♦ *AD FS 2.0 for Windows PowerShell Examples* (<http://technet.microsoft.com/en-us/library/adfs2-powershell-examples%28WS.10%29.aspx>)

## 10.5 Troubleshooting

There are several different sources to use for troubleshooting Access Manager problems:

- ♦ [“Troubleshooting the Administration Console”](#)

- ♦ [“Troubleshooting Certificate Issues”](#)
- ♦ [Chapter 17, “Troubleshooting Identity Server and Authentication,” on page 455](#)

---

# 11 Configuring Single Sign-On for Office 365 Services

NetIQ Access Manager provides single sign-on access to Office 365 services such as Exchange Server, Sharepoint Online and Lync without using ADFS (Active Directory Federation Services). You can use your existing enterprise credentials to access any of the Office 365 services without having to remember multiple passwords or sign in multiple times to access different services. You can sign in once with an existing password and Access Manager grants you access to all services.

This single sign-on access is achieved by implementing Passive or Active authentication by using WS-Federation, WS-Trust, and SAML 2.0 protocols.

A trust model is set up for Access Manager and Office 365 to communicate with each other. Access Manager, configured as an identity provider, allows Office 365 to trust it for authentication. Office 365 configured as a service provider, consumes authentication assertions from Access Manager.

- [Section 11.1, “Passive and Active Authentication,” on page 331](#)
- [Section 11.2, “Configuring Active and Passive Authentication By Using WS-Trust and WS-Federation Protocols,” on page 332](#)
- [Section 11.3, “Configuring an Office 365 Domain That Supports Passive Federation Using SAML 2.0 Protocol,” on page 335](#)
- [Section 11.4, “Useful Resources,” on page 339](#)
- [Section 11.5, “Troubleshooting Scenarios,” on page 340](#)
- [Section 11.6, “Sample Tokens,” on page 343](#)

## 11.1 Passive and Active Authentication

In a Passive authentication scenario, the user signs in through a Web form displayed by the identity provider and the user is requested to log in. In Active authentication scenario, the user is authenticated using thick clients. As the thick client does not support redirection, Office 365 gets the credentials and validates the authentication with Access Manager by communicating directly with it.

Passive authentication is supported by using the WS-Federation protocol and supports sign-in to Office 365 using the Web interface. The clients includes the Office 365 portal, SharePoint Online, Outlook Web Access, and the Office Web Apps. You can achieve passive authentication using either SAML 2.0 or WS-Federation protocol.

Active authentication is supported by using the WS-Trust protocol and supports sign-in to Office 365 using Office client applications. The clients includes Outlook, Lync, Word, Excel, PowerPoint, and OneNote. If you are using Microsoft Exchange, you can use SAML 2.0 but for active authentication, WS-Trust is the recommended protocol.

## 11.2 Configuring Active and Passive Authentication By Using WS-Trust and WS-Federation Protocols

Using the wizard, when you configure an Office 365 domain with WS-Trust protocol it creates the following two domains:

- ♦ A domain preconfigured for active authentication using WS-Trust protocol
- ♦ A domain preconfigured for passive authentication using WS-Federation protocol.

If your business needs demand using a domain based on SAML 2.0 protocol, you can configure a domain manually using the steps in [Section 11.3, “Configuring an Office 365 Domain That Supports Passive Federation Using SAML 2.0 Protocol,” on page 335](#)

The following sections cover the details about how to configure a domain by using WS-Trust and WS-Federation protocols:

- ♦ [Section 11.2.1, “Prerequisite,” on page 332](#)
- ♦ [Section 11.2.2, “Configuring an Office 365 Domain By Using WS-Trust Protocol,” on page 332](#)
- ♦ [Section 11.2.3, “Configuring an Office 365 Domain to Federate with Access Manager,” on page 333](#)

### 11.2.1 Prerequisite

Use the following steps to verify that WS-Trust and WS-Federation protocols are enabled in Access Manager:

- 1 In the Administration Console, click **Devices > Identity Servers > Edit**.
- 2 In the **Enabled Protocols** section, ensure that **WS-Trust** and **WS-Federation** protocols are selected.

### 11.2.2 Configuring an Office 365 Domain By Using WS-Trust Protocol

When you configure a new Office 365 domain by using the WS-Trust protocol, it creates a domain preconfigured for Active authentication and also creates a WS-Federation Service Provider that is preconfigured for Passive authentication.

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > WS-Trust > Service Provider Domain**.
- 2 Click **New > Office 365 Domain** and specify a name to identify the domain. This domain is by default configured with ImmutableID and Attribute Set information and a Service Provider with the same name as the Office 365 domain is automatically created.

An authentication method `Name/Password - Form-WebService` is created and this is selected for WS-Trust. This method ensures that an email address/password is accepted for authentication.

Click the domain name to make further modifications.

For more details, see link [Section 9.5.5, “Modifying Service Providers,” on page 305](#)

- 3 Click the **WS-Federation** tab and verify that a new Service Provider with the same name as the Office 365 domain is created. This Service Provider is preconfigured with Attribute Set information and Authentication Response for the Passive authentication.

## 11.2.3 Configuring an Office 365 Domain to Federate with Access Manager

- ♦ “Prerequisite” on page 333
- ♦ “Enabling Federation Settings in Office 365 Domain” on page 333
- ♦ “Verifying Single Sign-On Access” on page 334

### Prerequisite

Ensure that the following requirements are met before configuring an Office 365 domain:

- ♦ The Identity Server must be installed outside the firewall for it to access Office 365 domain.
- ♦ Sign up for an Office 365 account. For information about signing up, see [Sign in to Office 365](#).
- ♦ To single-sign on to any of the Office 365 applications, ensure that you download it from the Office 365 portal.
- ♦ Create a federated domain in Office 365 and prove ownership of it. This ensures that you add your company domain into the Office 365 domain. For more information, see [Adding and Verifying a Domain for Office 365](#).
- ♦ Ensure that the Windows 7 or Windows 8 workstations do not have the Active Directory Federation Service 2.0 snap-in installed.
- ♦ Ensure that the SSL certificate is issued by a well-known external certification authority (CA).
- ♦ If you are using Microsoft Lync and/or Microsoft Outlook thick clients with WS-Trust, replace the default self-signed SSL server certificate included with Access Manager with one that is signed by a public Certificate Authority (CA). This enables Office 365 to establish a trusted SSL session with Access Manager. For more information see, [Managing Trusted Roots and Trust Stores](#).

---

**NOTE:** If you are using Microsoft Lync, ensure that you enable federation. For more information, see [Lync External Access](#).

---

- ♦ Install Microsoft Live Sign-in Module to help manage and establish a remote session with the Office 365 account that is created to manage the Office 365 domain. To download, go to [Microsoft Downloads Center](#).
- ♦ Install Microsoft Azure Active Directory Module. To download, go to [Manage Azure AD using Windows PowerShell](#).

### Enabling Federation Settings in Office 365 Domain

Run the following commands in Powershell by modifying the commands with your domain name as per your setup. The domain name in the example is `namtest.com`.

- 1 From the Start Menu launch Windows Azure Active Directory Module for Windows PowerShell.
- 2 Run `$cred=Get-Credential`. Enter your cloud service administrator account credentials.
- 3 Run `Connect-MsolService -Credential $cred`

For example, if the name of the domain is `namtest.com` and the Base URL of the Identity Server is `https://namtest.com/nidp/`, execute the following commands at the Powershell prompt:

---

**NOTE:** In this example, the Base URL the port is not specified as it uses the default port 443. If you are using a different port, specify the port with the Base URL.

For example: `https://namtest.com/nidp/`

---

1. `$dom = "namtest.com"`
  2. `$url = "https://namtest.com/nidp/wsfed/ep"`
  3. `$ecpUrl = "https://namtest.com/nidp/wstrust/sts/active12"`
  4. `$uri = "https://namtest.com/nidp/wsfed/"`
  5. `$logouturl = "https://namtest.com/nidp/jsp/o365wsfedlogout.jsp"`
  6. `$mex = "https://namtest.com/nidp/wstrust/sts/mex"`
  7. `$cert = New-Object  
System.Security.Cryptography.X509Certificates.X509Certificate2("name and  
path of the certificate")`
- 

#### NOTE

- ♦ If the certificate used has a `.crt` extension ensure that you convert it to a `.cer` extension file.
  - ♦ While executing this command, ensure that you specify the path to the certificate within the double quotes. For example: `"C:\local\netiq-off365-sign.cer"`
- 

8. `$certData = [system.convert]::tobase64string($cert.rawdata)`

- 4 Use the following cmdlet to update the settings of the single sign-on domain.

```
Set-MsolDomainAuthentication -FederationBrandName $dom -Authentication  
Federated -PassiveLogOnUri $url -SigningCertificate $certData -IssuerUri $uri -  
ActiveLogOnUri $ecpUrl -LogOffUri $logouturl -MetadataExchangeUri $mex
```

## Verifying Single Sign-On Access

### Prerequisite:

- ♦ You need at least one user in Office 365 to verify that single sign-on is set up. If you have an existing user, ensure that the Immutable ID matches the GUID of the Access Manager user.

For instance if your user store is eDirectory and you want to retrieve the GUID of an existing Access Manager user, execute the following command on the eDirectory server terminal:

```
ldapsearch -D cn=<context> -w <password> -b <search base> cn=<fqdn of the  
administrator> GUID | grep GUID
```

Create an Office 365 user with this GUID as the Immutable ID using the following command in Powershell:

```
new-msolUser -userprincipalName "user1@domain name" -immutableID "GUID of  
user1" - lastname "lastname of user 1" -firstname user1 -DisplayName "user1  
users" -BlockCredential $false -"LicenseAssignment testdomain:ENTERPRISEPACK"  
-usageLocation "two letter country code[example: US,IN,DE,BE,GB etc]" -Password  
"password of the user" - LicenseAssignment validlicense.
```

### Procedure to verify:

To verify that single sign-on is set up correctly, perform the following procedure in a server that is not added to the domain.

- 1 Go to [Microsoft Online Services \(http://login.microsoftonline.com/\)](http://login.microsoftonline.com/)
- 2 Log in with your corporate credentials. (For example : `user1@namtest.com`)

If single sign-on is enabled, the password field is dimmed. You will instead see the following message: You are now required to sign in at <your company>.

- 3 Select the **Sign in at your company** link.

If you are able to sign in without errors, single sign-on is set up successfully.

## 11.3 Configuring an Office 365 Domain That Supports Passive Federation Using SAML 2.0 Protocol

- ♦ [Section 11.3.1, “Prerequisite,” on page 335](#)
- ♦ [Section 11.3.2, “Configuring an Office 365 Domain to Federate with Access Manager,” on page 335](#)

### 11.3.1 Prerequisite

Ensure that SAML 2.0 is enabled in Access Manager.

- 1 In the Administration Console, click **Devices > Identity Servers > Edit**.
- 2 In the **Enabled Protocols** section, verify whether SAML 2.0 is selected.

### 11.3.2 Configuring an Office 365 Domain to Federate with Access Manager

- ♦ [“Prerequisite” on page 335](#)
- ♦ [“Setting Up Office 365 Services” on page 336](#)
- ♦ [“Establishing Trust Between an Identity Provider and a Service Provider” on page 336](#)
- ♦ [“Configuring Desktop Email Client to Access Office 365 Emails” on page 337](#)
- ♦ [“Verifying Single Sign-On Access” on page 338](#)

#### Prerequisite

Ensure that the following requirements are met before configuring an Office 365 domain:

- ♦ Sign up for an Office 365 account. For information about signing up, see [Sign in to Office 365](#).
- ♦ To single-sign on to any of the Office 365 applications, ensure that you download it from the Office 365 portal.
- ♦ Create a federated domain in Office 365 and prove ownership of it. By doing this you add your company domain into the Office 365 domain. For more information, see [Adding and Verifying a Domain for Office 365](#).
- ♦ Ensure that the Windows 7 or Windows 8 workstations do not have the Active Directory Federation Service 2.0 snap-in installed.
- ♦ Ensure that the SSL certificate is issued by a well-known external certification authority (CA).
- ♦ If you are using Microsoft Lync and/or Microsoft Outlook thick clients with WS-Trust, replace the default self-signed SSL server certificate included with Access Manager with one that is signed by a public Certificate Authority (CA). This enables Office 365 to establish a trusted SSL session with Access Manager. For more information see, [Managing Trusted Roots and Trust Stores](#).

- ♦ Install Microsoft Live Sign-in Module to help manage and establish a remote session with the Office 365 account that is created to manage the Office 365 domain. To download, go to [Microsoft Downloads Center](#).
- ♦ Install Microsoft Azure Active Directory Module. To download, go to [Manage Azure AD using Windows PowerShell](#).

## Setting Up Office 365 Services

Office 365 is preconfigured to establish federation with an external service providers.

Perform the following steps to create a trusted service provider:

- 1 In the Administration Console, click **Devices > Identity Servers > Edit**.
- 2 Click **SAML 2.0 > New > Service Provider**.
- 3 Select **Provider Type** as `Office 365`. Ensure that **Source** is selected as the `Metadata Text`.
- 4 Specify a name for the Office 365 domain. The XML metadata is automatically populated in the **Text** field. Click **Next**.
- 5 Confirm the certificates and click **Finish** to save the changes.

## Establishing Trust Between an Identity Provider and a Service Provider

You can configure Office 365 domains federations by using the Microsoft Online Services Module. You can use the Microsoft Online Services Module to run a series of cmdlets in the Windows PowerShell command-line interface to add or convert domains for single sign-on.

Each Active Directory domain that you want to federate by using Access Manager must either be added as a single sign-on domain or converted to be a single sign-on domain from a standard domain. Adding or converting a domain sets up a trust between Access Manager and Office 365.

### Adding a Domain:

To add a domain to Office 365, perform the following steps:

- 1 Log in to Office 365 as an administrator.
- 2 On the Administrator page, click **Management > Domains > Add a domain**.
- 3 Specify the domain name that you want to add.
- 4 Click **Next**.
- 5 Verify the domain name.  
For more information about how to verify a domain, see [Verify your domain and change name servers](#).
- 6 Select appropriate services.
- 7 Configure the DNS records on the domain registrar for other services.

---

**NOTE:** Do not configure the new domain to the primary domain. Using the `Set-MsolDomainAuthentication` command to set the domain as a federated domain results in an error if the domain is the default domain.

---

For more information, see [Add a domain to Office 365](#).



**Converting a standard domain to a federated domain:** To convert a standard domain to a federated domain, perform the following steps:

- 1 Open the Microsoft Online Services Module from the Start menu.
- 2 Run `$cred=Get-Credential`. Enter your cloud service administrator account credentials.
- 3 Run `Connect-MsolService -Credential $cred`.

This cmdlet connects you to the cloud service. Creating a context that connects you to the cloud service is required before running any of the additional cmdlets installed by the tool.

For example, if the name of the domain you are converting to a single sign-on domain is *namtest.com*, and the base URL of the Identity Server is *https://namtest.com:8443/nidp*, execute the following commands at the Powershell prompt:

1. `$dom = "namtest.com"`
2. `$url = "https://namtest.com:8443/nidp/saml2/sso"`
3. `$ecpUrl = "https://namtest.com:8443/nidp/saml2/soap"`
4. `$uri = "https://namtest.com:8443/nidp/saml2/metadata"`
5. `$logouturl = "/nidp/jsp/o365Logout.jsp"`
6. `$cert = New-Object  
System.Security.Cryptography.X509Certificates.X509Certificate2("name and  
path of the certificate")`

---

**NOTE:** While executing this command, ensure that you specify the path to the certificate within the double quotes. For example: "C:\local\netiq-off365-sign.cer"

---

7. `$certData = [system.convert]::toBase64String($cert.rawdata)`

- 4 Use the following cmdlet to update the settings of the single sign-on domain:

```
Set-MsolDomainAuthentication -FederationBrandName $dom -Authentication Federated -  
PassiveLogOnUri $url -SigningCertificate $certData -IssuerUri $uri -ActiveLogOnUri $ecpUrl  
-LogOffUri $logouturl -PreferredAuthenticationProtocol SAML2
```

---

**NOTE:** Ensure that there are no spaces after the hyphen if you are copy pasting the command.

---

## Configuring Desktop Email Client to Access Office 365 Emails

You can configure your desktop email client to access Office 365 emails. The email clients must use a basic authentication and a supported exchange access method such as IMAP, POP, Active Sync, and MAPI.

The following are the list of email clients supported for this configuration:

- ♦ Microsoft Outlook 2007
- ♦ Microsoft Outlook 2010
- ♦ Thunderbird 8 and 9
- ♦ The iPhone (various iOS versions)
- ♦ Windows Phone 7

---

**NOTE:** You can download the email clients from the download section of Office 365.

---

These steps are explained with an example where the federated domain name is *namtest.com* and the base URL is *https://namtest.com:8443/nidp*. Replace the domain name and base URL based on your system configuration.

- 1 Open the Microsoft Online Services Module.

- 2 Run the following command:

```
$cred=Get-Credential
```

Specify your cloud service administrator account credentials.

- 3 Run the following command:

```
Connect-MsolService -Credential $cred
```

This cmdlet connects you to the cloud service.

- 4 Execute the following command to check the existing domain federation settings:

```
Get-MsolDomainFederationSettings -DomainName namtest.com
```

Substitute *namtest.com* with your domain name before executing this command.

In the output, look for the *ActiveLogOnUri* parameter.

For the Identity Server base URL *https://namtest.com:8443/nidp*, the value of the *ActiveLogOnUri* should be *https://namtest.com:8443/nidp/saml2/soap*. The *ActiveLogOnUri* is dependent on the base URL of the Identity Server.

If the value of *ActiveLogOnUri* in the command output is *https://namtest.com:8443/nidp/saml2/soap*, go to [Step 5](#) without modifying the configuration.

(Conditional) If the *ActiveLogOnUri* is not *https://namtest.com:8443/nidp/saml2/soap*, execute the following command. Substitute *namtest.com* and port *8443* with your domain name and port number respectively before executing the following command.

```
Set-MsolDomainFederationSettings -DomainName namtest.com -ActiveLogOnUri "https://namtest.com:8443/nidp/saml2/soap" -preferredauthenticationprotocol SAML2
```

- 5 Create a new email account in your email client and enter your Office 365 email ID.

---

**NOTE:** Configure Outlook related DNS settings before using email clients. You can configure these settings after adding the domain on the Office 365 port page.

---

- 6 The system prompts for specifying the basic authentication. Specify Access Manager credentials.

The email account is created after successful authentication.

---

**NOTE:** While logging in to the new email account, enter Access Manager credentials.

---

## Verifying Single Sign-On Access

You need at least one user in Office 365 to verify that single sign-on is set up. If you have an existing user, ensure that the Immutable ID matches with the GUID of the Access Manager user.

### Prerequisite:

- ♦ You need at least one user in Office 365 to verify that single sign-on is set up. If you have an existing user, ensure that the Immutable ID matches the GUID of the Access Manager user.

For instance, if your user store is eDirectory and you want to retrieve the GUID of an existing Access Manager user, execute the following command on the eDirectory server terminal:

```
ldapsearch -D cn=<context> -w <password> -b <search base> cn=<name of the user>  
GUID | grep GUID
```

Create an Office 365 user with this GUID as the Immutable ID using the following command in Powershell:

```
new-msolUser -userprincipalName "user1@domain name" -immutableID "GUID of  
user1" - lastname "lastname of user 1" -firstname user1 -DisplayName "user1  
users" -BlockCredential $false -"LicenseAssignment testdomain:ENTERPRISEPACK"  
-usageLocation "two letter country code[example: US,IN,DE,BE,GB etc]" -Password  
"password of the user".
```

If you want to use any other attribute as the ImmutableID of the Office 365 user, configure and then add a property name/value pair.

- 1 In the Administration Console, go to **Identity Server** and select an Identity Server.
- 2 Select **SAML 2.0** and then select the service provider you created.
- 3 Select **Options** and click **New**.
- 4 Add a property name/value pair as follows:

**Property Name:** SAML2\_OFFICE365\_NAMEID\_ATTRIBUTE\_NAME

**Property Value:** title

The title you specify in the **Property Value** should be base64 encoded and stored in the user store. This value should be used as ImmutableID while creating a user in Office 365.

### Verifying Single Sign-on:

To verify that single sign-on is set up correctly, perform the following procedure in a server that is not added to the domain:

- 1 Go to [Microsoft Online Services](#).
- 2 Log in with your corporate credentials. (For example : user1@nametest.com)  
If single sign-on is enabled, the password field is dimmed. You will instead see the following message: You are now required to sign in at <your company>.
- 3 Select the **Sign in at your company** link.  
If you are able to sign in without errors, single sign-on is set up successfully.

## 11.4 Useful Resources

The following list includes few useful resources for troubleshooting:

- ♦ [Office 365 Troubleshooting](#)
- ♦ [Microsoft Remote Connectivity Analyzer](#)
- ♦ [Sign in to Office 365 for Business](#)
- ♦ [Description of Office 365 Desktop Setup Tool Logging Errors](#)

### Links for Troubleshooting Sign-in issues with Microsoft Lync:

- ♦ [Describes how to troubleshoot sign-in issues with Microsoft Lync and Office 365](#)
- ♦ [Describes how to clear the credential cache that can occur on some operating systems](#)

- Describes how Microsoft Lync can cache its connection endpoints and how to resolve the issue
- Describes how to troubleshoot common issues between Lync for Mac and Office 365

## 11.5 Troubleshooting Scenarios

- Section 11.5.1, “WS-Trust and WS-Federation Scenarios,” on page 340
- Section 11.5.2, “SAML 2.0 Scenarios,” on page 341
- Section 11.5.3, “Office 365 Domain Scenarios,” on page 341

### 11.5.1 WS-Trust and WS-Federation Scenarios

#### Issue in Setting Up a Domain for Federation

If you try to set a primary domain for federation by running the `Set-MsolDomainAuthentication` command, it throws the following error:

`Set-MsolDomainAuthentication`: You cannot remove this domain as the default domain without replacing it with another default domain. Use the `Set-MsolDomain` cmdlet to set another domain as the default domain before you delete this domain.

To fix this issue, change the default domain by performing the following steps:

- 1 In the Office 365 portal, click **Organization Name** on the Admin page.
- 2 Click **Edit**.
- 3 Select a new default domain.

#### **Set-MsolDomainAuthentication : You cannot remove this domain as the default domain without replacing it with another default domain.**

If you get this error it indicates that you attempted to delete the default domain without replacing it with another domain.

Use the `Set-MsolDomain` cmdlet to set another domain as the default domain before you delete this domain.

#### **After upgrading iOS Apps to the Latest Version, Single Sign-On to Office 365 Services Fail**

To establish single sign-on from iOS apps to Office 365 services, perform the following steps:

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > Local > Contract**.
- 2 Specify a name to identify the contract.
- 3 Specify the URI as `http://schemas.microsoft.com/ws/2008/06/identity/authenticationmethod/password`.
- 4 Select **Name/Password - Form - WebService** method.

## 11.5.2 SAML 2.0 Scenarios

- ♦ [“SSO to Microsoft Services Fails” on page 341](#)
- ♦ [“Issue in Setting Up a Domain for Federation” on page 341](#)

### SSO to Microsoft Services Fails

SSO fails at Microsoft with this error:

```
Your organization could not sign you in to this service
```

Perform the following steps to fix this issue:

- ♦ Verify that the attributes are configured properly.

You can also use the SAML tracer plug-in Firefox to review the SAML assertion sent to Office365.

- ♦ Verify that federation settings are using the `Get-MsolDomainFederationSettings - DomainName <YOUR DOMAIN>` command.

### Issue in Setting Up a Domain for Federation

If you try setting up a primary domain for federation by running the `Set-MsolDomainAuthentication` command, it throws the following error:

```
Set-MsolDomainAuthentication: You cannot remove this domain as the default domain without replacing it with another default domain. Use the Set-MsolDomain cmdlet to set another domain as the default domain before you delete this domain.
```

To fix this issue, change the default domain by performing the following steps:

- 1 In the Office 365 portal, click **Organization Name** on the Admin page.
- 2 Click **Edit**.
- 3 Select a new default domain.

## 11.5.3 Office 365 Domain Scenarios

- ♦ [“Issues with the Directory Synchronization Tool” on page 341](#)
- ♦ [“Active Profile Authentication Fails for Microsoft Exchange Clients” on page 342](#)
- ♦ [“Microsoft Online Services Sign-In Assistant Installation Fails If Microsoft Office Professional Plus Is Installed” on page 342](#)
- ♦ [“Single Sign-On to Office 365 Domain Fails” on page 342](#)
- ♦ [“No License to Use Office 365 Services” on page 342](#)
- ♦ [“After Initial Successful Authentication, Unending Loop While Logging into Lync Using Wrong Username and Password” on page 342](#)

### Issues with the Directory Synchronization Tool

- ♦ If the installation of the Directory Synchronization tool fails, check the Event Viewer. Installation may fail if the Microsoft Online Service Sign-In Assistant is already installed on the system.

- If you require to uninstall the Directory Synchronization tool, log off and then login.
- If the Directory Synchronization tool is slow, increase RAM of the server.

## Active Profile Authentication Fails for Microsoft Exchange Clients

If the active profile authentication fails for Microsoft Exchange (Outlook) clients, verify that the necessary DNS records have been added to your DNS. For more information, see [Create DNS records at any DNS hosting provider for Office 365](#).

## Microsoft Online Services Sign-In Assistant Installation Fails If Microsoft Office Professional Plus Is Installed

Manually install Microsoft Online Services Sign-In Assistant, if its installation fails after installing Microsoft Office Professional Plus with this message:

"The Microsoft Online Services Sign In Assistant has experience an error. The error must be resolved before your subscription for this product can be verified. To retry subscription verification, first resolve error message 800704DD or try to manually install the Microsoft Online Services Sign In Assistant...."

You can download the installer from [MicroSoft Download Center](#).

After installation is complete, relaunch the service to verify your Office 365 license. For more information, see [Reactivate subscription license by using Osaui.exe](#).

## Single Sign-On to Office 365 Domain Fails

If single sign-on fails, ensure that the ImmutableID and the User Principal Name (UPN) matches the Office 365 user. To get Office 365 user details, log in to using Powershell and execute the following command:

```
Get-MsolUser -UserPrincipalName user1@namtest.com | fl *
```

## No License to Use Office 365 Services

If you receive an error stating that the user does not have license to use Office365, Log in to Office 365 as an administrator and assign required service licenses to the user.

## After Initial Successful Authentication, Unending Loop While Logging into Lync Using Wrong Username and Password

After successfully authenticating to Office 365 client, if you attempt to login to the Lync client using an incorrect username and password, Lync client uses the details from the previous successful session and tries to get a token from Access Manager. This results in an unending loop.

To resolve this issue, in the Lync client user interface, select the **Delete my sign-in info** option and log in once again.

## 11.6 Sample Tokens

- [Section 11.6.1, “Sample SAML Token,” on page 343](#)
- [Section 11.6.2, “Sample WS-Trust Token,” on page 345](#)
- [Section 11.6.3, “Sample WS-Federation Token,” on page 347](#)

### 11.6.1 Sample SAML Token

This section contains a sample XML for WS-Trust request and response.

#### Request:

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="_3ae4edbc-7ab5-48c7-a08e-
b8d6e395e02c" IssueInstant="2012-09-09T08:41:35Z" Version="2.0"
AssertionConsumerServiceIndex="0" ><saml:Issuer>urn:federation:MicrosoftOnline</
saml:Issuer><samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent" /></samlp:AuthnRequest>
```

#### Response:

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"

Destination="https://login.microsoftonline.com/login.srf" ID="idRuMHBv1VGqYUsw2Es-
SbA5Ue08w" InResponseTo="_3ae4edbc-7ab5-48c7-a08e-b8d6e395e02c"

  IssueInstant="2012-09-09T08:41:51Z" Version="2.0"><saml:Issuer>https://
www.netiqst.com/nidp/saml2/metadata</saml:Issuer><samlp:Status><samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success" /></samlp:Status><saml:Assertion
ID="idF5JceWGWYwS3bOkmJS2wJuNqitU" IssueInstant="2012-09-09T08:41:51Z"
Version="2.0"><saml:Issuer>https://www.netiqst.com/nidp/saml2/metadata</
saml:Issuer><ds:Signature xmlns:ds="http://www.w3.org/2000/09/
xmldsig#"><ds:SignedInfo><CanonicalizationMethod xmlns="http://www.w3.org/2000/09/
xmldsig#"
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" /><ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" /><ds:Reference
URI="#idF5JceWGWYwS3bOkmJS2wJuNqitU"><ds:Transforms><ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /><ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" /></
ds:Transforms><ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" /
><DigestValue xmlns="http://www.w3.org/2000/09/
xmldsig#">ZocFiEUycda0cKGRNcZYZqvmnlM=</DigestValue></ds:Reference></
ds:SignedInfo><SignatureValue xmlns="http://www.w3.org/2000/09/xmldsig#">
DLk4Uv/4VlwwKVz7XdQOdUv8ltcryLv2U3K7q57AE70wk/NNsa4kP8XdtA36Y470j+XTV+a+q0y
YsMNIEzySxaxMqo01Fm+6PfMH7HtTVj7fQ3n+VwANqbIs3G7eaaVlpHdUs79/dBujS8baNm1ZEBr
2gGVMWCHOa1fTOSZ08yPt9ume0PsYXpo2RdaoGkJCZUnVIIWg6UtI0zEKbY6mP3JhrUJ7OVHdbz
yNBzhfTv0m71nz0JKpy+i8MeDUIu1OiqTTIZ+c2SPceYhQcj8umrdE4JCGEBYNIIE52PalbRYgmLd
```

```

roAKn56vLDjq04VnYVRGhqP/McZwYZrx+7E7qQ==
</SignatureValue><ds:KeyInfo><ds:X509Data><ds:X509Certificate>
MIIFBzCCA++gAwIBAgIRAKdqzGh19tecryvMuy+QhgAwDQYJKoZIhvcNAQEFBQAwcjlEMakGA1UE
BhMCR0IxGzAZBgNVBAgTEkdyZWZ0ZXIqTWFuY2hlc3RlcjEQA4GA1UEBxMHU2FsZm9yZDEaMBGg
A1UEChMRQ09NT0RPIENBIExpbl0ZWQxGDAWBgNVBAMTD0Vzc2VudG1hbFNTTCBDQTAeFw0xMjA5
MDcwMDAwMDBaFw0xMjE5MDYyMzU5NTlaMFExITAFBgNVBAStGERvbWFPbiBD250cm9sIFZhbGlk
YXRlZDERMA8GA1UECXMIRnJlZSBTU0wxGTAXBgNVBAMTEHd3dy5uZXRpcXRzdC5jb20wgGEiMA0G
CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCX6k7wnFUoyPtqSj06xyQMHQtoXASBtHGASaOMxfZJ
rHQ4wbJUqMEtrxYcz4JxFrLzE8qvlY5r7cwxx/yvsiFwHq2HdRY6KU6I2u0eRF/tRwf3rl222/Xl
7wRbgdL43zd0yypjub9FKXlCxkaKucAlP+EVGtd7H8dFjMuf0iKZYvBFg9tcJWBGPfOw5iwe/rjK
6gQSXf13+Tpb6915lsusJfPMe3t04wA4XuyLlcJ/Jrxrj9xrEtwkmUcudTveZrVJfnz3NYXcW0J8
6a0JZSEiHlVHrIY/44fVEQFjkrfr2u5RKGBJzl35xb2x5mkUSzzy4CSL5p0fCsV0ve7LKx/fAgMB
AAGjggG3MIIBszAfBgNVHSMEGDAWgBTay+qtWwhdzP/8JlT0SeVVxjj0+DAdBgNVHQ4EFgQUEj/C
c5rqiBWiSzo9B8iJPdJnCpYwDgYDVR0PAQH/BAQDAgWgMAwGA1UdEwEB/wQCMAAwNAYDVR0lBC0w
KwYIKwYBBQUHAwEGCCsGAQUFBwMCBgorBgEEAYI3CgMBGlgkhkgBhvhCBAEwRQYDVR0gBD4wPDA6
BgsrBgEEAbIxAQICBzArMCKGCCsGAQUFBwIBFhlodHRwcZovL3NlY3VyZS5jb21vZG8uY29tL0NQ
UzA7BgNVHR8ENDAYMDCGlqAshipodHRwOi8vY3JsLmNvbW9kb2NhLmNvbS9Fc3NlbnRyYXwTU0xD
QS5jcmwmbG9YIKwYBBQUHAQEYjBgMDgGCCsGAQUFBzACHixodHRwOi8vY3J0LmNvbW9kb2NhLmNv
bS9Fc3NlbnRyYXwTU0xDQV8yLmNydDAKBggrBgEFBQcwAYYYaHR0cDovL29jc3AuY29tb2RvY2Eu
Y29tMCKGA1UdEQQiMCCCEHd3dy5uZXRpcXRzdC5jb22CDG5ldGlxdHN0LmNvbTANBgkqhkiG9w0B
AQUFAAOCAQEAJoS/fE0gBMVzQBzRRuSMBHMBnbgDXPlfVPwJZnkfIHbb/wXwYK7AqA5efOelAlqz
QD94kJ+W6JZm4ripePJk7QLnK2imqJb0E7LdmWQ3D05WQNsZKUklfR+9elP6xBN5ycXqtiEItScm
hE7H2gynz4/ejLXZv8XsBkfsYnT0wWUmyTsQYPLmVkJELfPiPGZsQcvpmSO9eoTQ8zabkQGjqzUM
NgGtXOMQBQgNO/7IMghgmSR0NduPguZoL3l0x84yKdf6Hl5cvbnH2W4c0n8vTkGcwUk80NYlTge
6TFPwzS98PzV08nxKSJWlhckasLQAYcw++bC7Blz+Nc7YyrNPw==
</ds:X509Certificate></ds:X509Data></ds:KeyInfo></
ds:Signature><saml:Subject><saml:NameID
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
NameQualifier="https://namtest.com:8443/nidp/saml2/metadata"
SPNameQualifier="urn:federation:MicrosoftOnline">bzM2NkBuZXRpcXRzdC5jb20=</
saml:NameID><saml:SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"><saml:SubjectConfirmationData
InResponseTo="_3ae4edbc-7ab5-48c7-a08e-b8d6e395e02c" NotOnOrAfter="2012-09-
09T09:41:51Z"
Recipient="https://login.microsoftonline.com/login.srf"/></
saml:SubjectConfirmation></saml:Subject><saml:Conditions NotBefore="2012-09-
09T05:55:12Z"

```



```

NotOnOrAfter="2012-09-09T11:28:30Z"><saml:AudienceRestriction><saml:Audience>...

SessionIndex="idF5JceWGWYwS3bOkmJS2wJuNqitU"><saml:AuthnContext><saml:AuthnContext
ClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password...

NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">

<saml:AttributeValue xsi:type="xs:string">o3662@netiqst.com</
saml:AttributeValue></saml:Attribute>

<saml:Attribute xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" Name="ImmutableID"...

</saml:AttributeValue></saml:Attribute></saml:AttributeStatement></
saml:Assertion></samlp:Response>

```

## 11.6.2 Sample WS-Trust Token

```

<saml:Assertion AssertionID="nsts150b8594-0aff-424f-8113-46045d943171"
IssueInstant="2014-05-09T07:00:18.019Z" Issuer="https://namnetiq.in/nidp/wsfed/"
MajorVersion="1" MinorVersion="1" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:excl4n="http://www.w3.org/2001/10/xml-exc-cl4n#"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:xs="http://www.w3.org/
2001/XMLSchema">
  <saml:Conditions NotBefore="2014-05-09T07:00:18.019Z" NotOnOrAfter="2014-05-
09T07:06:18.019Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>
        urn:federation:MicrosoftOnline
      </saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:Advice/>
  <saml:AuthenticationStatement AuthenticationInstant="2014-05-09T07:00:18.019Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
    <saml:Subject>
      <saml:NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified" NameQualifier="urn:federation:MicrosoftOnline">
        TLP1nEzIc0EEtEyz9ZxMyA==
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:bearer
        </saml:ConfirmationMethod>
      </saml:SubjectConfirmation>
    </saml:Subject>
  </saml:AuthenticationStatement>
  <saml:AttributeStatement>
    <saml:Subject>
      <saml:NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified" NameQualifier="urn:federation:MicrosoftOnline">
        TLP1nEzIc0EEtEyz9ZxMyA==
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:bearer
        </saml:ConfirmationMethod>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Attribute AttributeName="UPN" AttributeNamespace="http://
schemas.xmlsoap.org/claims">
      <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema">
        namtest@namnetiq.in
      </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="ImmutableID" AttributeNamespace="http://

```

```

schemas.microsoft.com/LiveID/Federation/2008/05">
  <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema">
    TLPInEzIc0EEtEyz9ZxMyA==
  </saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
shal"/>
    <ds:Reference URI="#nsts150b8594-0aff-424f-8113-46045d943171">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2000/09/
xmldsig#enveloped-signature"/>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>
        0Zvo3DbV0Qq7m9q7ER4Hol24bmA=
      </ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
    SqWAA39fYb3VJPBebZ6bsiUh0C+8ElbgDv2yG6xq3WLYUX/DoQ6RLfsb/1mVmMQBcGghUxhcDRAT
k6JA3djHbZCrZh7qblc8uBr+nm1Szps/BO7todTLu+g835WGSKdnpSoTjhO285MjsoomnrL+A4S
33F5Ld5OVOTPoarlwpBPFOgm7k9SnzjU0h7yIpp7YlZx1uF2sPvNeDRhkNEIsWwSPUY9mw04An9V
AsClCb1Q7+vEtCxxgJ4A6nxk8G9bvPRisk7H5fTihf0THNEzu5s6KnyGHCC6k2/jWHHF4Appg/aJ
ZelyQR9MKagNe60sAU2U83GM8Wust+o3+PvI3A==
  </ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>

MIIFSTCCBDGgAwIBAgIGb+MI39nZMA0GCSqGSIb3DQEBCwUAMIHGMQswCQYDVQQGEwJVUzEQMA4G
A1UECBMHQXJpem9uYTETMBEGA1UEBxMKU2NvdHRzZGFsZTElMCMGA1UEChMcU3RhcnZpZWxkIFRl
Y2hub2xvZ211cywgSW5jLjEzMDEGA1UECmMqAHR0cDovL2N1cnRzLnN0YXJmaWVsZHRlY2guY29t
L3JlcG9zaXRvcnkMTQwMgYDVQQDEytTdGFyZmllbGQgU2VjdXJlIElcnRzZmllYXRlIEF1dGhvc
cm10eSAtIEcyMB4XDTE0MDUwNjA5MDYwNVoXDTE1MDIyNjE5MDQwNFowOTEhMB8GA1UECmMYRG9t
YWluIEVnbvbnRyb2wgVmFsaWRhdGVkMRQwEgYDVQQDEwtuYWluZXRpcS5pbjCCASIdQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAMzjEinl0iWzMPKBQO+H2sb+HifrmVi7JDzhRfOKJakG+nXsgVx2
QRToN0UbvoeqldTaTZSKrFb0mc/E3aEkgSU67DazWvtm3nUSboJc4QVWQlJmXIP989K2H1DastwE
Srg6iW0MMUuz9ZadP3BQjV4VVB9qX81D321D4TilgYUDG5tpaUnftddiR+rZQR0ea3ABC0+oeZa
7w+jVVFUOAP+uG2iJ4zksIO+F3wIXDNZMYQwFlTvnCTO6/4cRWlXoGxh0BbZGdYn0qHzAOu9okT2B
gnz+aTaMGSIPPr+PXjB31XqeAhBRoXgrddWit1DawyrJETP0rzfMhdli+QsXHcCAwEAAAOCAccw
ggHDMaWGA1UdEwEB/wQCMAAwHQYDVVR01BBYwFAYIKwYBBQUHAWEGCCsGAQUFwMCMCA4GA1UdDwEB
/wQEAWIFoDA7BgNVHR8ENDAYMDCCgLAshipodHRwOi8vY3JsLnN0YXJmaWVsZHRlY2guY29tL3Nm
aWcyYzEtOC5jcmVwWQYDVVR0gBFiWUDBOBgtghkgBhvluAQcXATA/MD0GCCsGAQUFBwIBFjFodHRw
Oi8vY2VydGlmawNhdGVzLnN0YXJmaWVsZHRlY2guY29tL3JlcG9zaXRvcnkMTGCBggrBgEFBQcB
AQR2MHQwKgYIKwYBBQUHMAIGHmh0dHA6Ly9vY3NwLnN0YXJmaWVsZHRlY2guY29tLzBGBggrBgEF
BQcWAOY6aHR0cDovL2N1cnRzZmllYXRlcY5zdGFyZmllbGR0ZWNoLmNvbS9yZXBvc210b3J5L3Nm
aWcyLmNydDAfBgNVHSMEGDAwGBQlRYFoUCY4PTstLL7Natm2PbNmYzAnBgNVHREEIDAegggtuYWlu
ZXRpcS5pbjB0IPd3d3Lm5hbW5ldGlxLmLuMB0GA1UdDgQWBQANClv1YFFU3cAkVfQz/TxuttEUTAN
BgkqhkiG9w0BAQsFAAOCAQEAYsHcxqGpgrm9HSiSIFzD0dC9BraZdjh+fIUBeKRUBMsjsByPJIHj
OGuBnY8FtuPY8/elKhzhZcuUhY3zwVQzbWStWlraySJyO1SRRJC4onLbx42ARdKbRgxA/JDsmY
aTnyYq+ZOLm6XUtDweFEDkklAy2s08gru54ogJ0iD/JyX/dgZEH/v9lGjdNFUDwG4dLz++a2O1/U
UfqJye7Rb5UgNkewcG9KjydtGp7Mv6m8/JjzOl31ejIVVqgwz30fo+agirrIWWG2Ogtk0JUFrY73
coKTzspPszxMGN2FJpRSymtO+cgVlEuAK6/SCr2mhBvxg4GJuXuzSLp2kSrIfA==
    </ds:X509Certificate>
  </ds:X509Data>
</ds:KeyInfo>
</ds:Signature>
</saml:Assertion>

```

## 11.6.3 Sample WS-Federation Token

```
<wst:RequestedSecurityToken xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/
trust">
  <saml:Assertion AssertionID="idjTptEEQd5CuKy-0M-MBCY9lDHVQ"
IssueInstant="2014-05-09T06:44:07Z" Issuer="https://namnetiq.in/nidp/wsfed/"
MajorVersion="1" MinorVersion="1"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
    <saml:Conditions NotBefore="2014-05-09T06:29:07Z" NotOnOrAfter="2014-05-
09T06:59:07Z">
        <saml:AudienceRestrictionCondition>
            <saml:Audience>
                urn:federation:MicrosoftOnline
            </saml:Audience>
        </saml:AudienceRestrictionCondition>
    </saml:Conditions>
    <saml:AuthenticationStatement AuthenticationInstant="2014-05-09T06:44:07Z"
AuthenticationMethod="name/password/uri">
        <saml:Subject>
            <saml:NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">
                TLP1nEzIc0EEtEyz9ZxMyA==
            </saml:NameIdentifier>
            <saml:SubjectConfirmation>
                <saml:ConfirmationMethod>
                    urn:oasis:names:tc:SAML:1.0:cm:bearer
                </saml:ConfirmationMethod>
            </saml:SubjectConfirmation>
        </saml:Subject>
    </saml:AuthenticationStatement>
    <saml:AttributeStatement>
        <saml:Subject>
            <saml:NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">
                TLP1nEzIc0EEtEyz9ZxMyA==
            </saml:NameIdentifier>
            <saml:SubjectConfirmation>
                <saml:ConfirmationMethod>
                    urn:oasis:names:tc:SAML:1.0:cm:bearer
                </saml:ConfirmationMethod>
            </saml:SubjectConfirmation>
        </saml:Subject>
        <saml:Attribute AttributeName="UPN" AttributeNamespace="http://
schemas.xmlsoap.org/claims">
            <saml:AttributeValue>
                XX
            </saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute AttributeName="ImmutableID" AttributeNamespace="http://
schemas.microsoft.com/LiveID/Federation/2008/05">
            <saml:AttributeValue>
                XX
            </saml:AttributeValue>
        </saml:Attribute>
    </saml:AttributeStatement>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#" xmlns="http://www.w3.org/2000/09/xmldsig#" />
            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/
xmldsig#rsa-sha1" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
            <ds:Reference URI="#idjTptEEQd5CuKy-0M-MBCY9lDHVQ" xmlns:ds="http://
www.w3.org/2000/09/xmldsig#">
                <ds:Transforms xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:Transform Algorithm="http://www.w3.org/2000/09/
xmldsig#enveloped-signature" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                </ds:Transforms>
```



---

# 12 SP Brokering

This section describes the following brokering tasks:

- [Section 12.1, “Overview,” on page 349](#)
- [Section 12.2, “Configuring the SP Broker,” on page 351](#)
- [Section 12.3, “Configuring a Brokering for Authorization of Service Providers,” on page 354](#)
- [Section 12.4, “Creating and Viewing Brokering Groups,” on page 355](#)
- [Section 12.5, “Generating the Brokering URLs by Using an ID and Target in the Intersite Transfer Service,” on page 365](#)
- [Section 12.6, “Assigning The Local Roles Based On Remote Roles And Attributes,” on page 365](#)
- [Section 12.7, “SP Brokering Example,” on page 367](#)

## 12.1 Overview

The Identity Service acts as a Federation Gateway or a service provider broker (SP Broker). This feature is used along with the Intersite Transfer service of the identity provider which enables authentication at a trusted service provider.

The SP Broker helps companies establish trust between identity providers and their service providers that support different federation protocols. For example, an identity provider that supports SAML 2.0 can provide authentication to a Liberty or SAML 1.1 service provider by using the SP Broker.

The SP Broker helps reduce the number of trust relationships between an identity Provider and their service provider. For example, identity providers can now provide authentication to their service providers by establishing a single trust relationship instead of multiple trust relationships. Similarly, a service provider must establish a single trust relationship with the SP Broker to receive authentication from several identity providers.

The SP Broker feature helps control the authentication flow between several identity providers and service providers in a federation circle by allowing the administrator to configure policies that control Intersite Transfers. For example, an administrator can configure a policy with SP Broker that allows only certain users from an identity provider to be authenticated at a given service provider.

An Intersite Transfer URL has the following format: `https://<identity provider>/idpsend?PID=<Service Provider ID>&TARGET=<final_destination_URL>`

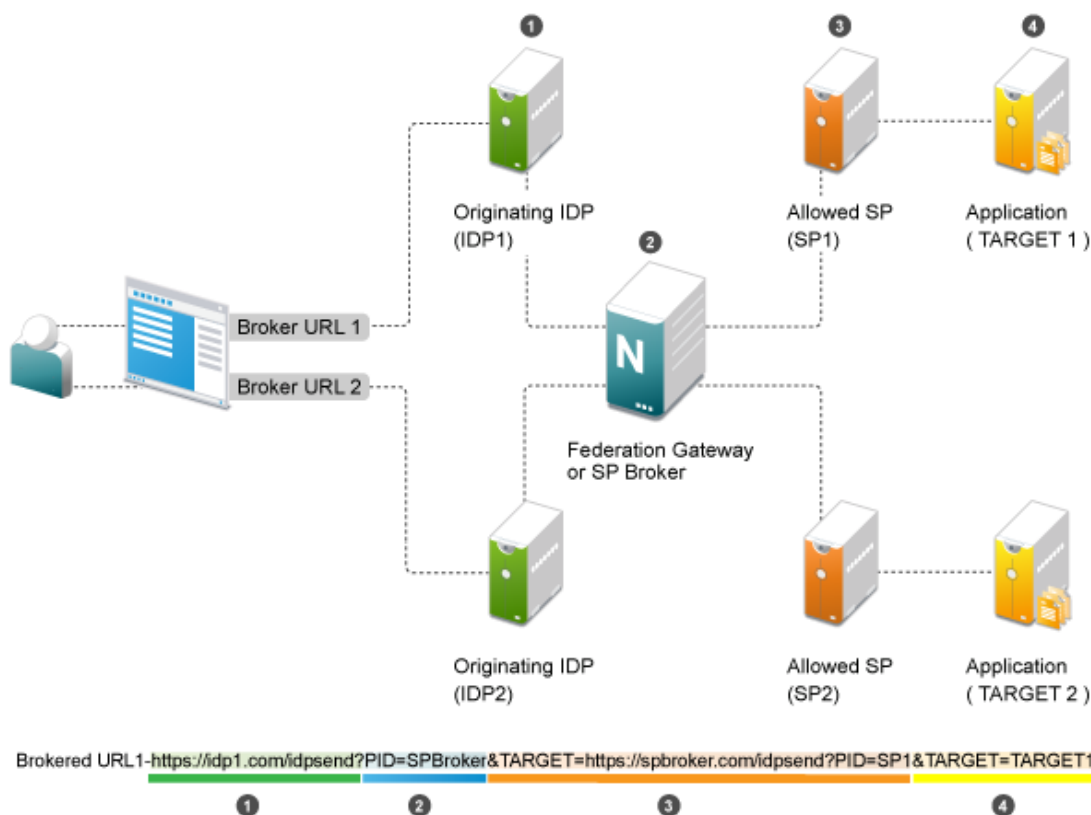
This Intersite Transfer URL consists of three parts:

- `https://<identity provider>`: The user can authenticate at the identity provider.
- `/idpsend?PID=<Service Provider ID>`: Authentication occurs at the service provider represented by the service provider ID at the identity provider.
- `&TARGET=<final_destination_URL>`: The user is finally redirected to the specified target URL associated with the service provider.

A Web Page is created with many Intersite Transfer URLs for each combination of identity provider, service provider, and the target application.

For more information about the Intersite Transfer Service, see [Section 7.18, “Using the Intersite Transfer Service,” on page 248.](#)

This following illustration explains the flow of providing access to the target URL by using SP Brokering:



**Web Page (User Portal):** A Web page (user portal) is created with a list of URLs called Brokered URLs, which provide access to various target applications.

**Originating Identity Providers:** The Originating Identity Provider is the identity provider with which the user credentials are stored for authentication. The Origin IDP must be configured as a Liberty/SAML1.1/SAML2.0 trusted identity provider in the SP Broker.

**Federation Gateway or SP Broker:** The Federation Gateway or SP Broker is a NetIQ identity provider that can be configured to control the authentication between several Origin IDPs and Allowed SPs in a federation circle.

**Allowed Service Provider:** The Allowed SP is the service provider in which the SP Broker provides authentication. The allowed SP must be configured as a Liberty/SAML1.1/SAML2.0 trusted service provider on SP Broker.

**Target Application:** The target application is the application running on a Web Server that is protected by the service provider.

**Broker URL:** A Broker URL is a specially designed Intersite Transfer URL, which consists of four parts. You can click the brokered URL, which results in the following:

1. You must authenticate with the Originating IDP (<https://idp1.com/idpsend>).
2. The Origin IDP causes an authentication to occur at the SP Broker ([?PID=SPBroker](https://spbroker.com/idpsend?PID=SPBroker)).

3. The SP Broker causes an authentication to occur at the allowed SP (TARGET=https://spbroker.com/idpsend?PID=SP1).
4. You are redirected to the target application (?TARGET=TARGET1).

SP brokering requests are the Intersite Transfers resulting from brokered URLs processed on the SP Broker. The SP Broker can control the brokering requests before providing an authentication to the service provider. The SP Broker enforces the policies configured by the administrator by either causing the authentication at the service provider or by denying the request.

The SP Broker provides the following options to configure policies that control SP brokering requests:

- 1 A set of SAML 1.1, SAML 2.0 and Liberty trusted identity providers and trusted service providers can be configured as a brokering group. The brokering request is allowed only if the Origin identity provider and Allowed service provider belong to the same brokering group. Brokering Request is not allowed from an Origin identity provider of one group to an Allowed service provider of another group.
- 2 In a brokering group, a set of brokering rules can be configured that provides granular control on the brokering requests. For example, a brokering rule can be configured to deny a brokering request from an Origin identity provider to an Allowed service provider, if the user satisfies a certain condition at the SP Broker.

SP brokering is enabled on the Identity Server only if at least one brokering group is enabled. If an Intersite Transfer request is received with neither the origin identity provider nor the Allowed service provider in any of the brokering group, the request is treated as a regular Intersite Transfer and SP brokering controls are not applied.

## 12.2 Configuring the SP Broker

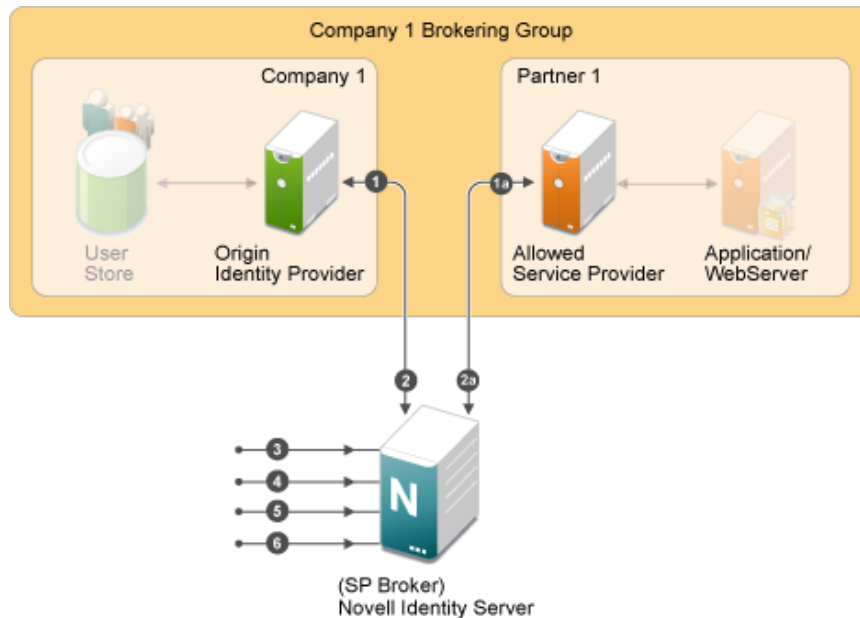
This section describes how to configure the origin identity provider to act as the SP Broker or a Federation hub and also control authentications provided by the Origin identity providers to their Allowed service providers.

### Prerequisites

1. Identify the Origin identity providers and their Allowed service providers. For example, Company 1 establishes a business partnership with Partner 1, at which Company 1 users can access the application of Partner 1. In this case, identity provider at Company 1, is now the Origin identity provider and Allowed service provider is the service provider at Partner 1, and controls access to its applications.
2. Identify the federation protocols supported by the Origin identity providers and their Allowed service providers. NetIQ Identity Provider supports SP brokering for SAML 2.0, Liberty, or SAML 1.1 federations.
3. Identify whether Persistent or Transient federations needs to be established between Company 1 and Partner 1. For Persistent federation, the user that is authenticated at the Origin identity provider must be mapped to a valid user at their Allowed service provider. For Transient federation, the user is provided with a temporary identity at the Allowed service provider.

## Configuration Flow

The following diagram depicts the various configuration steps involved in enabling the SP Brokering feature assuming SP Brokering is enabled between Company 1 and Partner 1:



### Step 1: Establish Federation at Origin Identity Providers to SP Broker

1. The SP Broker must be configured as the service provider at origin identity provider of Company 1.

For more information, see [Section 7.3.2, “Creating a Trusted Service Provider for SAML 1.1 and Liberty,”](#) on page 212.

If NetIQ identity provider is the Origin identity provider, refer.

For more information, see [Section 7.3.1, “Creating a Trusted Service Provider for SAML 2.0,”](#) on page 210.

### Step 1a: Establish Federation at Allowed Service Providers to SP Broker

The SP Broker must be configured as the identity provider at the allowed service provider of Partner 1.

For more information, see [Section 7.3.3, “Creating a Trusted Identity Provider,”](#) on page 214.

If NetIQ identity provider is the allowed service provider, refer.

For more information, see [Section 7.3.1, “Creating a Trusted Service Provider for SAML 2.0,”](#) on page 210.

---

**NOTE:** Step 1 must be repeated for each of the origin identity provider in the federation circle and step 1a must be repeated for each of the allowed service provider in the federation circle.

---



## Step 2: Establish Federations at SP Broker for Origin Identity Providers and their Allowed Service Providers

At the SP Broker, configure origin identity provider of Company 1 as the identity provider. The federation protocol (SAML 1.1/ SAML 2.0/ Liberty) and the federation type (Persistent or Transient) must match the federation protocol and federation type that is used for the respective origin identity provider in the step 1 above.

For more information, see [Section 7.3.2, “Creating a Trusted Service Provider for SAML 1.1 and Liberty,” on page 212.](#)

For more information, see [Section 7.3.1, “Creating a Trusted Service Provider for SAML 2.0,” on page 210.](#)

### Step 2a:

At the SP Broker, configure the allowed service provider of Partner 1 as the service provider. The federation protocol (SAML 1.1/ SAML2.0/ Liberty) and the federation type (Persistent or Transient) must match the federation protocol and the federation type that is used for the respective allowed service provider in the Step1a above.

For more information, see [Section 7.3.3, “Creating a Trusted Identity Provider,” on page 214.](#)

For more information, see [Section 7.3.1, “Creating a Trusted Service Provider for SAML 2.0,” on page 210.](#)

---

**NOTE:** Step 2 must be repeated for each of the origin identity provider in the federation circle and Step 2a must be repeated for each of the allowed service provider in the federation circle.

---

## Step 3: Configure the Attribute to be Cached at the SP Broker (Optional)

If the target applications require user information, then this information must be passed along with the authentication by the origin identity provider. At the SP Broker, these attributes that are received at the authentication must be cached and sent to the allowed service provider during authentication.

For more information, see [Section 7.8.1, “Configuring the Attributes Obtained at Authentication,” on page 224.](#)

For more information, see [Section 7.8.2, “Configuring the Attributes Sent with Authentication,” on page 225.](#)

## Step 4: Create Brokering Group for the Federation Circle with the Origin Identity Providers and their Allowed Service Providers

Create a new brokering group in the SP Broker. Company 1 identity provider and Partner 1 service provider must be selected as the origin identity provider and their allowed service provider.

For more information, see [Section 12.4.1, “Creating a Brokering Group,” on page 356.](#)

The SP Brokering is enabled when at least one brokering group is enabled. Origin identity providers and their allowed service providers can either be added while creating the brokering group or added/ deleted by editing the brokering group.

## Step 5: Create Brokering Rules for the Brokering Group

Create brokering rules that provide granular control on the brokering requests. For example, a brokering rule can be configured which can deny a brokering request from an origin identity provider to an Allowed service provider, if the user satisfies a certain role at the SP Broker.

For more information, see [Section 12.4.3, “Configuring Brokering Rules,” on page 359](#).

To use roles in the brokering rules, identity role policies must be configured on NetIQ identity provider acting as an SP Broker. Roles can be associated for the user at the SP Broker according to the various parameters that include roles sent by the origin identity provider, attribute values sent from the identity provider.

For more information, see [Section 1.2, “Enabling Role-Based Access Control,” on page 28](#).

Several rules can be configured for a brokering group. To help administrators understand how the rules are applied, a Rule Validation user interface is provided under each brokering group.

For more information, see [Section 12.4.5, “Validating Brokering Rules,” on page 363](#).

## Step 6: Create and Configure Brokering URLs

The users at Company 1 are provided with a portal page containing URLs to access the applications at the service provider of Partner 1. These URLs are called Brokering URLs and are designed to pass through the SP Broker. The URLs consists of information that are embedded and a tool is provided to construct these URLs.

The administrator can create URLs from a given origin identity provider to their allowed service provider to access a given target application specified by a target URL. The URLs constructed are placed in the users' portal page.

For more information, see [Section 12.4.4, “Constructing Brokering URLs,” on page 361](#).

## 12.3 Configuring a Brokering for Authorization of Service Providers

Authorization rules for authorizing service provider requests must be configured from the Access Manager Brokering page. To configure authorization policy, configure the broker rule policy. Ensure that the service providers are configured to the local Identity Server that will be evaluated during authorization. [Figure 12-1 on page 355](#) displays the sample configuration.

Figure 12-1 SAML2 Service Provider Initiated Authorization Rule Configuration

### Edit the Brokering Rule

Rule Name   
 Rule Priority

**Trusted Providers**

**Origin IDP**

☐ Any IDP  
☒ The following:

Allowed IDPs:   
 Available Trusted IDPs in the Group:

**Allowed SP**

☐ Any SP  
☒ The following:

Allowed SPs:   
 Available Trusted SPs in the Group:

**Role Conditions**

[New](#) | [Delete](#)

☐ Condition  
☐ brokerrule

**Action**

☐ Permit ☒ Deny

OK Cancel Apply

## 12.4 Creating and Viewing Brokering Groups

The identity server cluster configuration provides a **Brokering** tab that you can use to configure the groups and generate brokered URLs.

- Section 12.4.1, “Creating a Brokering Group,” on page 356
- Section 12.4.2, “Configuring Trusted Identity Providers and Service Providers,” on page 357
- Section 12.4.3, “Configuring Brokering Rules,” on page 359
- Section 12.4.4, “Constructing Brokering URLs,” on page 361
- Section 12.4.5, “Validating Brokering Rules,” on page 363

- 1 In the Administration Console, click **Devices > Identity Servers > Brokering**.
- 2 The **Brokering** tab allows you to create new Groups as well as display the configured Groups. The Display Brokering Groups page displays the list of groups configured. You can also create, delete, enable, and disable the brokering group on this page.

General					Local	Liberty	SAML 1.1	SAML 2.0	WS Federation	Brokering	WS-Trust
New   Delete   Enable   Disable											
<input type="checkbox"/>	Name	Enabled	Identity Providers	Service Providers	Brokering Rules						
Brokering Groups											
<input type="checkbox"/>	<a href="#">Group1-1</a>		2	2	LibertytoSAML2,...(3)						

3 The Display Brokering Groups page displays the following information for each group:

**Group Name:** Specifies a unique name to identify the group. When you click on the hyperlink, you can view the Group Details page, where the Group configuration such as name and list of Identity Providers and Service Providers can be modified.

**Enabled:** A check mark indicates that brokering is enabled for the group by applying the configured rules. A blank means that brokering is disabled.

**Identity Providers:** Display the total number of Liberty/SAML1.1/SAML2 IDPs assigned to this group.

**Service Providers:** Display the total number of Liberty/SAML1.1/SAML2 SPs assigned to this group.

**Brokering Rules:** If the rules are not configured, then “No Rules Config” is displayed. The default rule allows for brokering between any IDP to any SP in the group. If new rules are configured, then the first rule name is displayed along with the count of total rules.

## 12.4.1 Creating a Brokering Group

When a brokering group is created while grouping the brokering feature, following rules are applicable:

- ♦ Brokering is not allowed among different company groups.

The brokering is not allowed between the logical customers of Company 1 Brokering Group and Company 2 Brokering Group.

- ♦ Brokering is allowed among different partners of the company group.

Brokering is allowed between the brokering groups of Company 1 Brokering Group and Company 2 Brokering Group.

- ♦ Role based brokering is allowed among Company 1 and Partner 1 logical customers.
- ♦ Role based brokering is allowed among Company 2 and Partner 2 logical customers.
- ♦ Brokering is allowed among different partners based on roles and groups authentication of the company.

To create a new broker group follow these steps:

- 1 In the Administration Console, click **Devices > Identity Servers > Brokering**.
- 2 Click **New**. The Creating Brokering Group page displays.
- 3 Perform the following actions in the fields:

**Create Brokering Group**

**Configure Trusted Providers**

Display name:

**Selected IDPs**

idp\_company1

**Available Trusted IDPs**

idp\_company2  
idp\_partner1A  
idp\_partner1B  
idp\_partner2A

**Selected SPs**

sp\_company1

**Available Trusted SPs**

sp\_company2  
sp\_partner1A  
sp\_partner1B  
sp\_partner2A

<< Back   Finish   Cancel

**Display Name:** Specify the brokering group display name.

**Selected IDPs:** Select at least one trusted IDP using navigation button.

**Selected SPs:** Select at least one trusted SP using navigation button.

**Available Trusted IDPs:** Displays Liberty/SAML1.1/SAML2.0 trusted IDP configured on the given IDP cluster (idp\_cluster1).

**Available Trusted SPs:** Displays Liberty/SAML1.1/SAML2.0 Trusted Service Providers configured on the given Identity Provider Cluster (idp\_cluster1).

- 4 Click **Finish** to complete creation of the brokering group creation.

## 12.4.2 Configuring Trusted Identity Providers and Service Providers

You can configure the rules between the trusted identity providers and service providers by configuring rules, roles, and actions. You can view the configured rules, create new, delete the existing rule, edit the rules, enable and disable the configured rules.

You can configure the service providers and identity providers for all of the protocols in the Identity Server, which are configured in the identity server cluster. Using the brokering group, you can view the list of available service providers and identity providers in the selection box. Using the arrow keys, configure the trusted identity providers and trusted service providers for the respective brokering group.

**Configuration**

**Trusted Providers** | Rules | Construct URL | Rule Validation

Display name:

Selected IDPs		Available Trusted IDPs
122company2_idp 130logincompany1	← →	123partner1b 127partner2b_idp Company1_130_iDFF_IDP Company1_130_saml1_IDP

Selected SPs		Available Trusted SPs
127partner2b_sp	← →	122company2_SP 123partner1b 130logincompany1 Company1_130_saml1_SP

OK Cancel Apply

- 1 In the Administration Console, click **Devices > Identity Servers > Brokering Group Name**. The Configuration page displays the **Trusted Providers**, **Brokering rules**, **Construct URL** and **Rule Validation** tabs.
- 2 Click **Trusted Providers** tab.
- 3 Specify the display name and configure the brokering groups.
 

**Display Name:** Specify the display name of the configuring brokering group.

**Select IDPs:** Configure the selected identity providers using the arrow keys from the available trusted IDPs.

**Available Trusted IDPs:** Configure the available trusted identity providers using the arrow keys from **Selected Identity Providers** selection box.

**Selected SPs:** Configure the selected service providers using the arrow keys from the **Available Trusted Service Providers** selection box.

**Available Trusted SPs:** Configure the available trusted service providers using the arrow keys from the **Selected Service Providers** selection box.
- 4 Click **OK** to continue and the configured service providers and identity providers details are displayed in the Brokering page.
- 5 Click **Finish** to complete the rules configuration for the brokering group.
- 6 Click **Apply** to see the configuration changes.

**NOTE:** When you log out from the Access Gateway device, then the logout is not propagated on the other Identity Servers if you have SAML 1.1 as one of the trusted provider in the brokering group.

## 12.4.3 Configuring Brokering Rules

You can create, edit, delete, enable, and the disable brokering rules.

- 1 In the Administration Console, click **Devices > Identity Servers > Brokering**.
- 2 Click the existing or newly created Brokering Group hyperlink.
- 3 Click **Rules**. The Brokering Group Rules page is displayed.

Configuration

Trusted Providers | Rules | Construct URL | Rule Validation

New | Delete | Enable | Disable

<input type="checkbox"/> Name	Enabled	Identity Providers	Service Providers	Priority	Action	Role Conditions
Brokering Rules						
<input type="checkbox"/> <a href="#">DENY-Manager-</a>		1	1	1	Deny	! MANAGER,...(2)
<input type="checkbox"/> <a href="#">CEO</a>		3	1	1	Deny	CEO(1)
<input type="checkbox"/> <a href="#">DENY-EMP</a>		3	1	1	Deny	EMP(1)
<input type="checkbox"/> <a href="#">Not-Allow-Manager-from-IDP2</a>		1	1	1	Deny	MANAGER(1)
<input type="checkbox"/> <a href="#">DENY SPBROLE</a>		3	1	1	Deny	SPBROLE(1)
<input type="checkbox"/> <a href="#">HIGH-RULE</a>		Any	Any	1	Permit	No Role Conditions Configured
<input type="checkbox"/> <a href="#">RULE1</a>		Any	Any	1	Permit	No Role Conditions Configured
<input type="checkbox"/> <a href="#">RULE-MANAGER</a>		3	1	1	Permit	! MANAGER(1)
<input type="checkbox"/> <a href="#">ALLOW-ALL</a>		Any	Any	10	Permit	No Role Conditions Configured

OKCancelApply

**Name:** Displays the rule name of the brokering group.

**Enabled:** Displays the status of the brokering group rule.

**Identity Providers:** Displays the number of identity providers configured to the brokering group.

**Service Providers:** Displays the number of service providers configured to the brokering group.

**Priority:** Displays the brokering group rule priority number.

**Actions:** Displays the configured brokering group rule action status either as permit or deny.

**Role Conditions:** Displays the brokering group role condition, such as manager and employee , configured on the rule page.

- 4 Click **OK** to continue and display the configured brokering group rule details on the Brokering Rules page.
- 5 Click **Apply** to see the brokering rule configuration changes.

## Creating a Brokering Rule

You can configure the rules to the created brokering groups.

- 1 In the Administration Console, click **Devices > Identity Servers > Brokering**.
- 2 Click the existing or newly created Brokering Group hyperlink.

3 Click **Rules**. The Creating Brokering Group page displays.

**Rule Name:** Specify the name of the rule.

**Rule Priority:** Select the rule priority from the drop-down list.

---

**NOTE:** The default rule specified during creation of the group has a priority of 1. Additional rules can be added, and existing rules can be deleted or modified. You can use the Edit Rules Page to modify the priority of the rules.

---

**Origin IDP:** Displays all Identity Servers or one or more Identity Servers that are available in the group.

**Allowed SP:** Displays all service providers or one or more service providers that are available in the group.

**Role Conditions:** Displays the brokering group role condition such as manager and employee, configured on the rule page.

**Actions:** Select the Permit or Deny action radio button for the rule you configure to the brokering group.

---

**NOTE:** By default, Access Manager allows any role. If you want to allow access to only particular roles, configure a permit condition for roles with higher priority and configure a deny condition in which no roles are defined with lower priority.

---

4 Click **Finish** to complete configuration of rules for the brokering group.



## Deleting a Brokering Rule

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > Brokering > (Brokering Group in the Brokering Group list) > Rules**.
- 2 Select the check box of the brokering group rule you want to delete, then click **Delete**. A message is displayed as “Delete selected brokering rule(s)?”.
- 3 Click **OK** to continue.

## Enabling a Brokering Rule

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > Brokering > (Brokering Group in the Brokering Group list) > Rules**.
- 2 Select the check box of the brokering group rule you want to enable.
- 3 Click **Enable**. The selected brokering group is enabled.

## Disabling a Brokering Rule

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > Brokering > (Brokering Group in the Brokering Group list) > Rules**.
- 2 Select the check box of the brokering group you want to disable from the brokering group rule configuration.
- 3 Click **Disable**. The selected brokering group is disabled.

## Editing Brokering Rules

You can edit the group rules in the Brokering page.

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > Brokering**.
- 2 Click the existing or newly created brokering group hyperlink.
- 3 Click **Rules** tab.
- 4 Click the Brokering Rules hyperlink to edit the information. The Edit Brokering Rule page displays the information. You can also edit the information.

You can edit all the fields and modify the information about the Create Brokering Rule page. For more information about create brokering rule, see [“Creating a Brokering Rule” on page 359](#)

## 12.4.4 Constructing Brokering URLs

The Construct URL page helps you to create a URL, which you use in your application to navigate to your trusted partners.

You can generate the URL according to the origin and allowed service provider Identity Servers.

- 1 In the Administration Console, click **Devices > Identity Servers > Brokering**.
- 2 Click the existing or newly created brokering group hyperlink.
- 3 Click **Construct URL**.

**Configuration**

Trusted Providers | Rules | **Construct URL** | Rule Validation

**Origin IDP**

IDP Type: Local IDP ▼

Origin IDP: 130logincompany1 ▼

OriginIDP URL:

Provider Param Name: PID

Target Param Name: TARGET

Allowed SP: 127partner2b\_sp ▼

Target URL:

Login URL: `https://SPBROKER1.labs.blr.novell.com:8443/nidp/saml2/idpsend?PID=https%3A%2F%2Flogin.partner2B.ccm%3A8443%2Fnidp%2Fsaml2%2Fmetadata&TARGET=<TargetURL>`

**IDP Type:** Select the Identity Provider type from the drop-down list. The three types of IDP in the drop-down list are Local IDP, NetIQ IDP, and Other IDP. If you select NetIQ IDP as the IDP type, then you can select the Origin IDP from the drop-down list. If you select Other IDP as the IDP type, you can enter the Origin IDP URL and you can select the Origin IDP from the drop-down list.

**Origin IDP:** The Origin identity providers are the trusted providers. The drop-down list displays all the trusted providers created for the specific NetIQ brokering group. Select the Origin IDP from the drop-down list.

---

**NOTE:** If the Origin IDP drop-down list does not list any trusted providers, it is because a local Identity Server exists as a trusted provider. To resolve this, add another Identity Server to the NetIQ brokering group

---

**Origin IDP URL:** If you select Other IDP as the IDP type, you can enter the Origin IDP URL manually. The <OriginIDPURL> represents (protocol :// domain : port / path ? querystring).

**Provider Parameter Name:** If you select Other IDP as the IDP Type, you can enter the trusted provider parameter ID. For more information about Intersite Transfer Service target for a service provider, see [Section 7.18.4, “Configuring an Intersite Transfer Service Target for a Service Provider,” on page 254](#)

**Target Parameter Name:** If you select Other IDP as the IDP type, you can enter the target provider parameter name manually.

**Allowed SP:** The allowed service providers are the selected service providers of the trusted providers. The drop-down list displays all the service providers created for the specific brokering group. Select the service providers from the drop-down list.

**Target URL:** Specify the target URL for the specific trusted providers and service provider pair. This URL will be appended to the login URL. Click **Generate** to generate the login URL

**Login URL:** The login URL consists of Origin IDP URL and the target URL.

- 4 Click **Cancel** to close the Construct URL page.

## 12.4.5 Validating Brokering Rules

The rule validation page helps you to validate the Origin identity providers and the allowed service provider rule according to the role associated with the respective trusted partners.

- 1 In the Administration Console, click **Devices > Identity Servers > Brokering**.
- 2 Click on the existing or newly created brokering group hyperlink.
- 3 Click the **Rule Validation** tab.



The screenshot shows a web interface for configuring a brokering rule. At the top, there is a tabbed menu with four tabs: "Configuration" (selected), "Trusted Providers", "Rules", "Construct URL", and "Rule Validation". Below the tabs, there are three input fields: "Origin IDP" with a dropdown menu showing "122company2\_idp", "Allowed SP" with a dropdown menu showing "127partner2b\_sp", and "Role" with a text input field containing "employee". At the bottom of the form is a button labeled "Validate Rule".

**Origin IDP:** The Origin identity providers are the trusted providers. The drop-down list displays all the trusted providers created for the specific NetIQ brokering group. Select the Origin identity providers from the drop-down list.

**Allowed SP:** The Allowed SPs are the selected SPs of the trusted providers. The drop-down list displays all the service providers created for the specific brokering group. Select the service providers from the drop-down list.

**Role:** Specify the role you want to validate for the selected Origin identity trusted providers and allowed SP. Click the Validate Rule.

A list is displayed according to the rule validation for the selected trusted providers, role, and permission.

Configuration						
Trusted Providers   Rules   Construct URL   Rule Validation						
<div>  Permit </div>						
Name	Identity Providers	Service Providers	Priority	Action	Role Conditions	Evaluate State
DENY-Manager-	130logincompany1	127partner2b_sp	1	Deny	! MANAGER(1)	Ignored
CEO	122company2_idp 130logincompany1 Local IDP	127partner2b_sp	1	Deny	CEO(1)	Disabled
DENY-EMP	122company2_idp 130logincompany1 Local IDP	127partner2b_sp	1	Deny	EMP(1)	Disabled
Not-Allow-Manager-from-IDP2	122company2_idp	127partner2b_sp	1	Deny	MANAGER(1)	Disabled
DENY SPBROLE	122company2_idp 130logincompany1 Local IDP	127partner2b_sp	1	Deny	SPBROLE(1)	Disabled
HIGH-RULE	Any	Any	1	Permit	No Role Conditions Configured	Disabled
<div>Cancel</div>						

**Name:** Displays the role name of the selected trusted providers.

**Identity Providers:** Displays the identity provider name.

**Service Providers:** Displays the service provider name.

**Priority:** In ascending order, displays the priority number of the rule validation of the selected trusted providers.

**Action:** Displays the permission action for validation of the selected trusted providers rule validation.

**Role Conditions:** Displays the role conditions for the selected trusted providers rule validation. Denial takes precedence over Permit.

**Evaluate State:** Displays the role conditions evaluate state for the selected trusted providers rule validation. You can see different evaluation states in the role conditions.

**Pass 1:** If the rule matches the Origin identity provider, allowed service provider or any roles mentioned.

**Pass2:** If the rule matches the Origin identity provider, allowed service provider or any specific role mentioned.

**Ignored:** If the rule does not match either Pass 1 or Pass 2 .

**Not Executed:** The default state of all the roles.

---

**NOTE:** If the rule has the evaluate State as Pass 1 action as Deny, then the remaining rules are in the non-executed state.

After a rule has the evaluate state as Pass 2, regardless of the action, the remaining rules are in the non-executed state.

The rules before Pass 1, should have the evaluate state of Ignored. All these ignored rules should have the role condition as **Any**, without specifying any role condition.

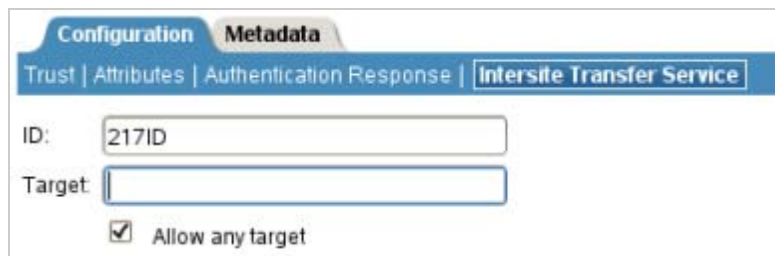
Pass 1 evaluation stops, as soon as a match for the Origin identity provider and allowed service provider is found with specific to some role condition.

- 
- Click **Cancel** to close the Rule Validation page.

## 12.5 Generating the Brokering URLs by Using an ID and Target in the Intersite Transfer Service

You can generate the brokering URL's using the ID of the target. You can use this value to simplify the Intersite Transfer Service URL that must be configured at the service provider. For more information, see [Section 7.18.4, "Configuring an Intersite Transfer Service Target for a Service Provider,"](#) on page 254.

- 1 In the Administration Console, click **Devices > Identity Servers > Brokering** or click **Devices > Identity Servers > Edit > SAML 2.0 > Trusted Providers > (Broker Identity under the Service Providers list) > Intersite Transfer Service**.



- 2 **ID:** Specify the ID value of the target.
- 3 **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. For example, if you enter the target URL as it appears below, then it will be displayed when you select **Allow Any Target** option.

```
https://login.company1.com:8443/nidp/saml2/
idpsend?id=217ID&TARGET=https%3A%2F%2FSPBROKER1.labs.blr.novell.com%3A8443%2Fn
idp%2Fsaml2%2Fidpsend%3FPID%3Dhttps%3A%2F%2Flogin.partner2B.com%3A8443%2Fnidp%
2Fsaml2%2Fmetadata%26TARGET%3Dhttps%3A%2F%2Fpartner2b.com
```

- 4 **Allow any Target:** Select this option to 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 you can see the URL specified in the **Target** option.

## 12.6 Assigning The Local Roles Based On Remote Roles And Attributes

You are able to configure the attributes based on the roles you select in the Attribute set field. You are able to log in and authenticated based on roles federated in the Origin Identity Provider, Target Service Provider and the Brokering Service Provider configuration.

### Origin Identity Provider Role Attribute Configuration

- 1 In the Administration Console, click **Devices > Identity Servers > Shared Settings > Attribute Sets > Mapping > New**. The **Add Attribute Mapping** window displays.
- 2 Select the local attribute name from the drop-down list
- 3 Enter the remote attribute name for the selected local attribute.
- 4 Click **OK** to add the remote attribute name. The newly added attribute displays in the Mapping list.

- 5 In the Administration Console, click **Devices > Identity Servers > Edit > SAML 2.0 > Trusted Providers > (Broker Identity under the Identity Providers list) > Configuration > Attributes**.
- 6 Select the role from drop-down list in the **Attribute set**.
- 7 Using the arrows map the attributes in the **Send with Authentication** and **Available List**.
- 8 Click **Apply** to map the set role and attribute of the origin Identity Provider.

## Target Identity Provider Role Attribute Configuration

- 1 In the Administration Console, click **Devices > Identity Servers > Shared Settings > Attribute Sets > Mapping > New**. The **Add Attribute Mapping** window displays.
- 2 Select the local attribute name from the drop-down list
- 3 Enter the remote attribute name for the selected local attribute.
- 4 Click **OK** to add the remote attribute name. The newly added attribute displays in the Mapping list.
- 5 In the Administration Console, click **Devices > Identity Servers > Edit > SAML 2.0 > Service Providers > (Broker Identity under the Service Providers list) > Configuration > Attributes**.
- 6 Select the role from drop-down list in the **Attribute set**.
- 7 Using the arrows map the attributes in the **Send with Authentication** and **Available List**.
- 8 Click **Apply** to map and set the attribute changes to the selected role of the target Identity Service Provider.

## Brokering Service Provider Role Attribute Configuration

The roles set and the attribute configured in origin identity provider and the target service provider is added and mapped in the brokering service provider attribute configuration.

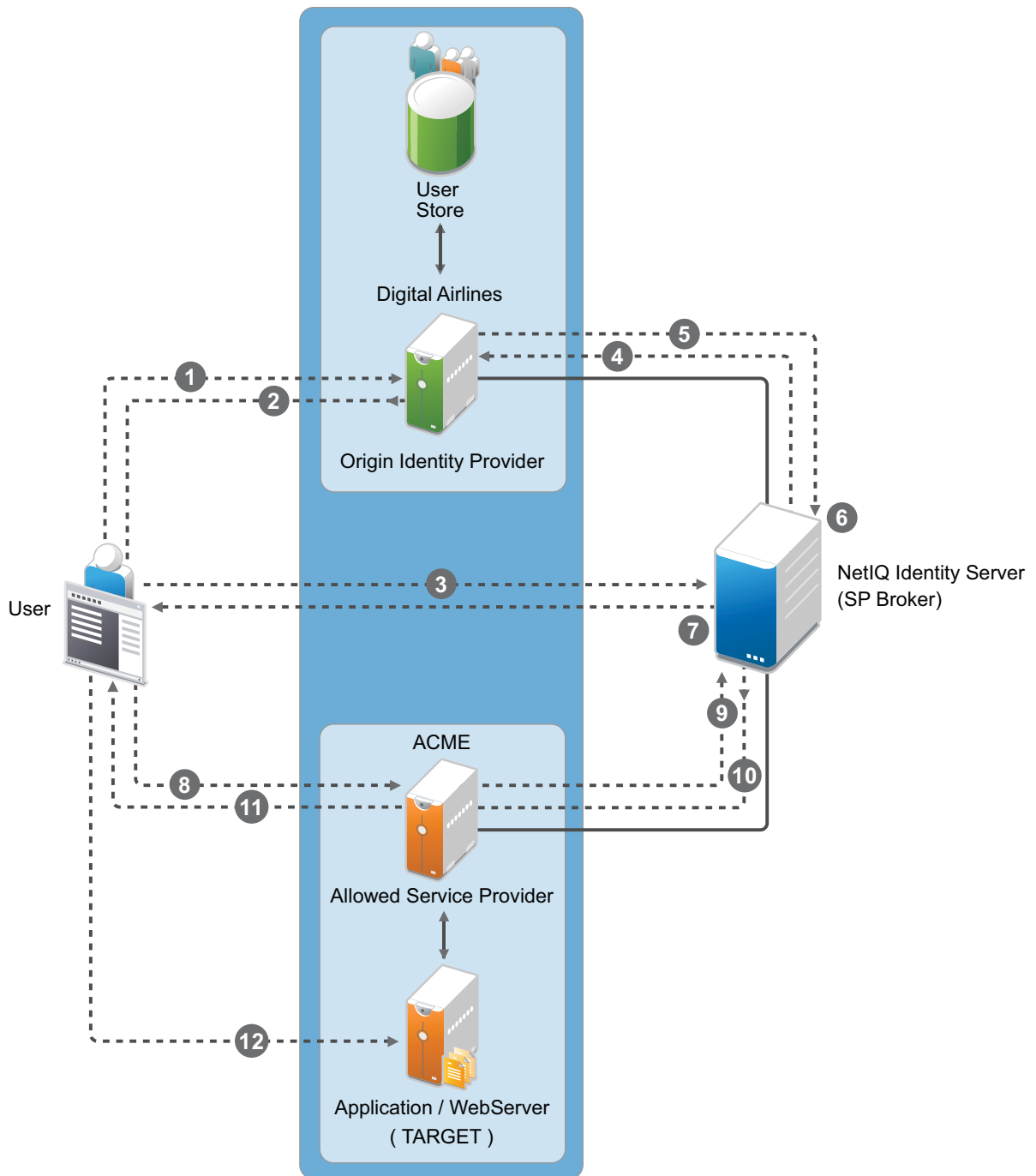
- 1 In the Administration Console, click **Devices > Identity Servers > Shared Settings > Attribute Sets > Mapping > New**. The **Add Attribute Mapping** window displays.
- 2 Select the local attribute name from the drop-down list
- 3 Enter the remote attribute name for the selected local attribute.
- 4 Click **OK** to add the remote attribute name. The newly added attribute displays in the Mapping list.
- 5 In the Administration Console, click **Devices > Identity Servers > Brokering** or click **Devices > Identity Servers > Edit > SAML 2.0 > Service Providers > (Broker Identity under the Service Providers list) > Configuration > Attributes**.
- 6 Select the role from drop-down list in **Attribute set**.
- 7 Using the arrows map the attributes in **Send with Authentication** and **Available List**.
- 8 Click **Apply** to set the role and configure the attribute mappings.

## 12.7 SP Brokering Example

This example explains how SP Brokering works. Let us assume that two companies Digital Airlines and ACME are business partners. There are certain applications that users of both Digital Airlines and ACME require to access.

With SP Brokering, users in Digital Airlines are provided with an intersite transfer URL that allows users to authenticate at Digital Airlines, set the assertion at ACME, and give access to the target application. With this approach, users do not have to choose from different authentication cards.

The following diagram depicts the SP Brokering workflow:



**Workflow:**

1. A user is authenticated at Digital Airlines identity provider. The user clicks Broker URL. Digital Airlines checks if this user is authenticated. If not, it asks for user credentials and authenticates the user.
2. Digital Airlines identity provider processes an intersite URL and creates an assertion for SP Broker (NetIQ Identity Server).
3. SP Broker receives the assertion and validates that this assertion is received from a trusted identity provider.
4. SP Broker checks if the trusted identity provider and the service provider (available in the target URL) belong to the same group. SP Broker denies the request if both do not belong to same group.
5. SP Broker sends a request to Digital Airlines identity provider to resolve the artifact.
6. SP Broker receives the SAML assertion from Digital Airlines identity provider and caches attributes/roles received. SP Broker applies any Role policies that have been enabled.
7. SP Broker performs intersite transfer. In the processing of intersite transfer, SP Broker checks if this user was a result of SP Brokering (step 4 earlier). SP Broker enforces the SP Brokering rules check: if any of the rules result in deny, an error page is displayed.
8. SP Broker creates an assertion for ACME.
9. ACME sends a request to SP Broker to resolve the artifact.
10. ACME receives the SAML assertion from the SP Broker along with roles/attributes.
11. ACME sends a redirect to the final target URL. (Note: Redirect happens from ACME's ESP to ACME's identity provider where the user is already authenticated.)
12. The user accesses the target application.



---

# 13 Configuring User Identification Methods for Federation

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:

- [Section 13.1, “Defining User Identification for Liberty and SAML 2.0,” on page 369](#)
- [Section 13.2, “Defining User Identification for SAML 1.1,” on page 373](#)
- [Section 13.3, “Defining the User Provisioning Method,” on page 375](#)
- [Section 13.4, “User Provisioning Error Messages,” on page 379](#)

## 13.1 Defining User Identification for Liberty and SAML 2.0

- [Section 13.1.1, “Selecting a User Identification Method for Liberty or SAML 2.0,” on page 369](#)
- [Section 13.1.2, “Configuring the Attribute Matching Method for Liberty or SAML 2.0,” on page 372](#)

### 13.1.1 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 7.10, “Configuring an Authentication Request for an Identity Provider,” on page 232](#)), 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**.

Configuration

Metadata

Authentication Card

Trust | Attributes | **User Identification** | Options

User Identification Methods

☒ **Authenticate**  
☐ Allow 'Provisioning'

☐ **Provision account**

☐ **Attribute matching**  
☒ Prompt for password on successful match  

Provisioning settings (undefined)

Attribute Matching settings (undefined)

Post Authentication methods:

Selected methods:

(Note: Select the methods which don't require user identification)

Available Methods:

Name/Password - Basic  
Name/Password - Form  
Secure Name/Password - Basic  
Secure Name/Password - Form

☐ Logout on method execution failure

☐ Allow IDP to set session timeout  
☐ Overwrite Temporary User  
☐ Overwrite Real User  

Assertion Validity Window  seconds

OK

Cancel

Apply

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.

---

**NOTE:** If you have selected Transient user mapping while configuring the name identifier format, the **Authenticate** and **Provision account** options are not displayed for SAML 2.0.

---

**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 13.1.2, "Configuring the Attribute Matching Method for Liberty or SAML 2.0," on page 372](#).
- ♦ If you selected the **Provision account** option, select a method, then click **OK**. If you have not created a provisioning method, continue with [Section 13.3, "Defining the User Provisioning Method," on page 375](#).
- ♦ 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 13.3, "Defining the User Provisioning Method," on page 375](#).
- ♦ If you selected the **Authenticate** option without the **Allow Provisioning** option, click **OK**.

**4** Configure the post authentication method.

**Selected Methods:** Using the arrow keys to move methods from the **Available Methods** list to the **Selected Methods** list. The selected method is executed when post remote authentication completes.

For example if you select the passwordfetch method, this method is executed at the service provider after the identity provider authentication and federation completes.

**Logout on method execution failure:** If you select this check box, then whenever there is a session failure, the user is logged out automatically.

**5** Configure the session options.

**Allow IDP to set session timeout:** Select Allow Identity Provider to set session timeout between the principal identified by the subject and the SAML authority based on **SessionNotOnOrAfter** attribute in SAML assertion of **authnStatement**.

**Overwrite Temporary User:** If you select this check box, then the temporary user credentials profile got from previous authentication method in the same session will be overwritten with real user credentials profile got from this authentication method.

**Overwrite Real User:** If you select this check box, then the real user credentials profile got from previous authentication method in the same session will be overwritten with real user credentials profile got from this authentication method

**Assertion Validity Window:** You can manually set the assertion validity time for SAML Service Provider (SP) to accommodate clock skew between Service Provider and SAML Identity (IDP) Server.

**6** Click **OK** twice, then update the Identity Server.

## 13.1.2 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 15.2, “Managing Web Services and Profiles,” on page 386](#).

- 1 In the Administration Console, click **Devices > Identity Servers > Servers > Edit > Liberty [or SAML 2.0] > [Identity Provider] > User Identification**.
- 2 Click **Attribute Matching settings**.

Identity Servers ► sp-401k ► Corporate IDP ►

### User Matching Method



Select User Stores to search

User stores:	Available user stores:
Installed User Store	

↑ ↓

User Matching Expression: <Select User Matching Expression>

If match not found: Do nothing

- 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 about creating a look-up expression, see [Section 6.3, “Configuring User Matching Expressions,” on page 194](#).
- 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 7.11, “Configuring an Authentication Response for a Service Provider,” on page 237](#).

---

- ♦ **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 about this process, see [Section 13.3, “Defining the User Provisioning Method,” on page 375](#).
- 8 Click **OK** twice, then update the Identity Server.

## 13.2 Defining User Identification for SAML 1.1

- [Section 13.2.1, “Selecting a User Identification Method for SAML 1.1,” on page 373](#)
- [Section 13.2.2, “Configuring the Attribute Matching Method for SAML 1.1,” on page 374](#)

### 13.2.1 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**.

The screenshot shows the 'User Identification' configuration page. At the top, there are tabs for 'Configuration', 'Metadata', and 'Authentication Card'. Below these are sub-tabs for 'Trust', 'Attributes', 'User Identification' (which is selected), and 'Options'. A dropdown menu labeled 'Satisfies contract:' is set to 'Name/Password - Form'. Below this is a section titled 'User Identification Methods' with two radio button options: 'Do nothing' (which is selected) and 'Attribute matching'. Under 'Attribute matching', there is a checked checkbox for 'Prompt for password on successful match'. Below that, 'Attribute Matching settings' is shown with a pencil icon and the text '(undefined)'. At the bottom, 'Assertion Validity Window' is set to '300' seconds. At the very bottom are three buttons: 'OK', 'Cancel', and 'Apply'.

- 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.

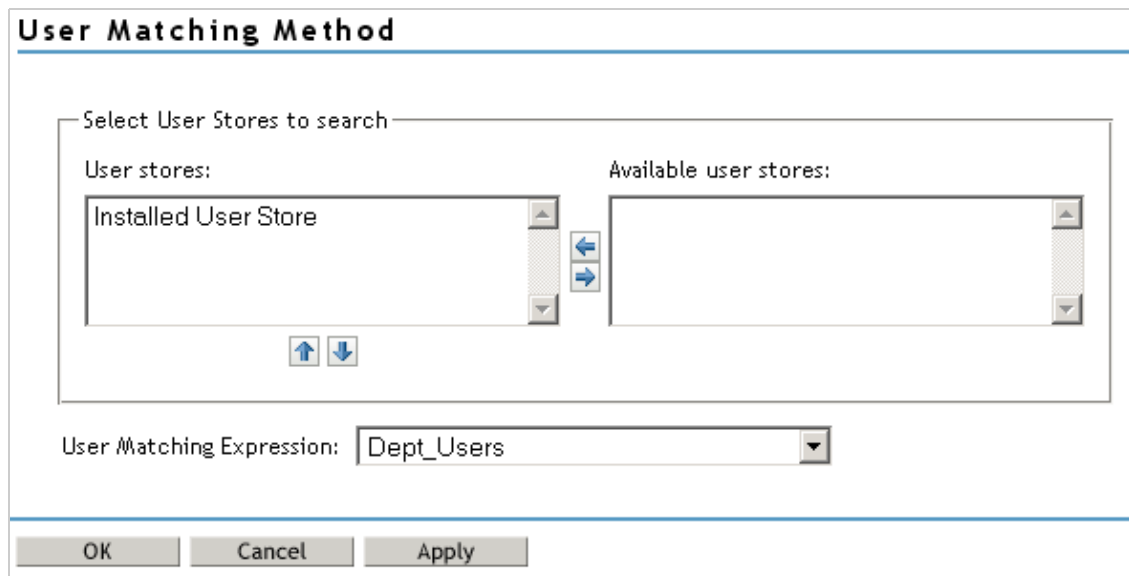
- 3 Select one of the following options for user identification:
  - ♦ **Do nothing:** Specifies that an identity provider account is not matched with a service provider account. This option allows the user to authenticate the session without identifying a user account on the service provider.
  - ♦ **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 6](#).
  - ♦ If you selected **Attribute matching**, continue with [Section 13.2.2, “Configuring the Attribute Matching Method for SAML 1.1,” on page 374](#).
- 5 You can also configure the assertion time manually.
  - ♦ **Assertion Validity Window:** You can manually set the assertion validity time for SAML Service Provider (SP) to accommodate clock skew between Service Provider and SAML Identity (IDP) Server.
- 6 Click **OK** twice.
- 7 Click **Apply** to make the user identification configuration changes.
- 8 Update the Identity Server.

## 13.2.2 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.

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 15.2, “Managing Web Services and Profiles,” on page 386](#).

- 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**.



The dialog box is titled "User Matching Method". It contains a section labeled "Select User Stores to search" which includes two list boxes: "User stores:" and "Available user stores:". The "User stores:" list box contains the entry "Installed User Store". Between the two list boxes are two blue arrows pointing in opposite directions. Below the "User stores:" list box are two small blue arrows pointing up and down. Below the "Available user stores:" list box are two small blue arrows pointing up and down. At the bottom of the dialog box is a label "User Matching Expression:" followed by a dropdown menu showing "Dept\_Users". At the very bottom are three buttons: "OK", "Cancel", and "Apply".

- 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 about creating a look-up expression, see [Section 6.3, "Configuring User Matching Expressions," on page 194](#).
- 5 Click **OK**.
- 6 Update the Identity Server.

## 13.3 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 15.6, "Mapping LDAP and Liberty Attributes," on page 397](#).

---

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.

Identity Servers > sp-401k > Corporate IDP >

### User Provisioning Method ?

**Step 1 of 5: Select required attributes**

Required attributes must exist on the service provider when creating a new user account, or the provisioning request fails and the user account is not created. The available attributes are standard Liberty Alliance attributes.

Attributes:	Available attributes:
Informal Name	Every Day Name
Job Title	Common Personal Title
Department Name	Common First Name
	Common Last Name
	Common Middle Name
	Legal Name
	Legal Personal Title
	Legal First Name
	Legal Last Name
	Legal Middle Name
	Legal Fiscal Identification Type
	Legal Fiscal Identification Value
	Date of Birth
	Gender
	Marital Status
	Portrait Image URL
	Home Page URL
	Name Pronounced Audio File URL
	My Greeting for Others Audio File URL
	How I Want to be Greeted Audio File URL

Navigation: Up, Down

- 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.

Identity Servers > sp-401k > Corporate IDP >

### User Provisioning Method ?

**Step 3 of 5: Define user name creation**

Selecting an attribute for the user name segments from the required attributes list will improve the chances the new user name will be created.

Maximum length:  character(s)

☒ **Prompt for user name**

☐ **Automatically create user name**

Segment 1:  Length:  character(s)

Junction:

Segment 2:  Length:  character(s)

☐ Ensure name is unique



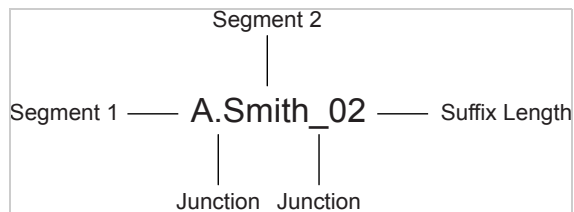
You can specify whether users are prompted to create their own usernames or whether the system automatically creates usernames. Selecting an attribute for the username segments from the required attributes list improves the chances that a new username is successfully created.

**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.



Identity Servers > sp-401K > Corporate IDP

### User Provisioning Method ?

**Step 4 of 5: Define new user password creation**

The new user account will not be valid after the initial use if the user is not given the generated password.

Min. password length:   

Max. password length:   

☐ Prompt for password

☒ Automatically create password

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.

Identity Servers > sp-401K > Corporate IDP

### User Provisioning Method ?

**Step 5 of 5: Select User Store where new user account is created**

The selected User Store will be the target directory. Specify the directory context where the new user accounts will be created.

User Store:

Context:  (ex. ou=users,o=novell)

☐ Delete user provisioning accounts if federation is terminated

**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.

## 13.4 User Provisioning Error Messages

The following error messages are displayed for the end user if there are problems during provisioning:

**Table 13-1** 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.	<p>The provisioned user account was deleted without first defederating the user. Remove orphaned identity objects from the configuration datastore.</p> <p><b>IMPORTANT:</b> 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.</p>
Unable to complete authentication request.	<p>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.</p> <p>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.</p>



---

# 14 Configuring Communication Profiles

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.

- ♦ [Section 14.1, “Configuring a Liberty Profile,” on page 381](#)
- ♦ [Section 14.2, “Configuring a SAML 1.1 Profile,” on page 382](#)
- ♦ [Section 14.3, “Configuring a SAML 2.0 Profile,” on page 382](#)

## 14.1 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.

## 14.2 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.

The Post method is enabled by default and you are not able to modify the default settings. The Post profile creates a metadata that includes only a Post binding on the Service Provider. If you have the default setup, then always both Artifact and Post options are enabled. If both the options are enabled, then by default Artifact binding is used. If Artifact binding is disabled or removed, only Post method is used.

**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.

## 14.3 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.

---

**NOTE:** If **Show logged out providers** option is enabled with HTTP Post profile, logout request from service provider does not complete. This is due to the difference in the HTTP method used in the logout request. It is recommended to use HTTP Redirect method when **Show logged out providers** option is enabled.

---

**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.



---

# 15 Configuring Liberty Web Services

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:

- ♦ [Section 15.1, "Configuring the Web Services Framework," on page 385](#)
- ♦ [Section 15.2, "Managing Web Services and Profiles," on page 386](#)
- ♦ [Section 15.3, "Configuring Credential Profile Security and Display Settings," on page 394](#)
- ♦ [Section 15.4, "Customizing Attribute Names," on page 396](#)
- ♦ [Section 15.5, "Configuring the Web Service Consumer," on page 396](#)
- ♦ [Section 15.6, "Mapping LDAP and Liberty Attributes," on page 397](#)

For additional resources about the Liberty Alliance specifications, visit the [Liberty Alliance Specification \(http://www.projectliberty.org/resources/specifications.php\)](http://www.projectliberty.org/resources/specifications.php) page.

## 15.1 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**.

## 15.2 Managing Web Services and Profiles

After a service has been discovered and 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 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 15.2.4, “Editing Web Service Policies,” on page 391](#).

**Edit a profile:** To edit a profile, click the name of a profile. For information about configuring the details, see [Section 15.2.1, “Modifying Service and Profile Details for Employee, Custom, and Personal Profiles,” on page 387](#) and [Section 15.2.2, “Modifying Details for Authentication, Discovery, LDAP, and User Interaction Profiles,” on page 389](#).

For information about modifying the description, see [Section 15.2.3, “Editing Web Service Descriptions,” on page 390](#).

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 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.

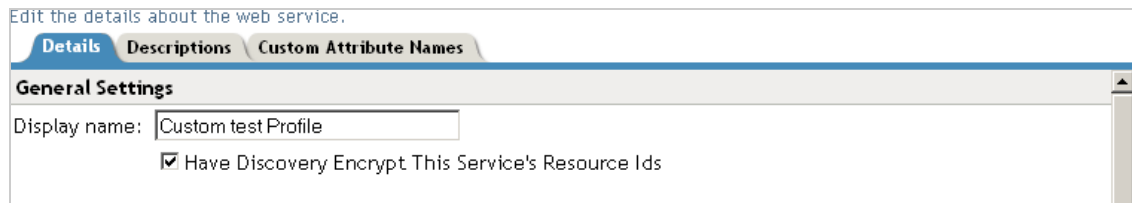
**User Interaction:** Allows you to set up a trusted user interaction service, used for identity services that must interact with the resource owner to get information or permission to share data with another Web service consumer. This profile enables a Web service consumer and Web service provider to 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.

## 15.2.1 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).



Edit the details about the web service.

Details Descriptions Custom Attribute Names

**General Settings**

Display name:

☒ Have Discovery Encrypt This Service's Resource Ids

- 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:

**Custom Profile**

Edit the details about the web service.

**Details** | **Descriptions** | **Custom Attribute Names**

---

**General Settings**

Display name:

☐ Have Discovery Encrypt This Service's Resource Ids

---

**Data Locations Settings**

Selected Read Locations:

- Configuration Datastore
- LDAP Data Mappings
- Remote Attributes

Available Read Locations:

Selected Write Locations:

- LDAP Data Mappings
- Configuration Datastore

Available Write Locations:

---

**Data Model Extensions**

Extend the service data model by defining new data types.

---

OK Cancel Apply

**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 15.6, “Mapping LDAP and Liberty Attributes,” on page 397](#).
- ♦ **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 15.6, “Mapping LDAP and Liberty Attributes,” on page 397](#).

**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 487](#) for more information.

7 Click **OK**, then click **OK** on the Web Service Provider page.

8 Update the Identity Server.

## 15.2.2 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**.

## 15.2.3 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 decisions.
  - ♦ **SAML:** Used for message exchanges that generally rely upon message authentication as well as the conveyance and attestation of 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.

## 15.2.4 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.

### Personal Profile Policy

Service Policy Categories	
6 Item(s)	
Name	
<a href="#">All Trusted Providers</a>	
<a href="#">Owner</a>	
<a href="#">Trusted Service Provider: 10.10.157.30</a>	
<a href="#">Trusted Service Provider: 10.15.167.56</a>	
<a href="#">Trusted Service Provider: ag40_group_LAG</a>	
<a href="#">Trusted Service Provider: nag_group_NAG</a>	

- 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**.



## Owner

All Service Policy			
Edit Policy▼			1 Item(s)
<input type="checkbox"/> Policy	Modify Policy	Inherited	
<input type="checkbox"/> Entire Personal Identity	Ask Me	Ask Me : Ask Me	
<div> <div>Query: Ask me</div> <div>Query: Always Allow</div> <div>Query: Never Allow</div> <div>.....</div> <div>Modify: Ask me</div> <div>Modify: Always Allow</div> <div>Modify: Never Allow</div> <div>.....</div> <div>Query and Modify: Ask me</div> <div>Query and Modify: Always Allow</div> <div>Query and Modify: Never Allow</div> </div>			

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.

- 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)
<input type="checkbox"/> Policy	Query Policy	Modify Policy	Inherited
<input type="checkbox"/> Informal Name	Ask Me	Ask Me	Ask Me : Ask Me
<input type="checkbox"/> Localized Informal Name	Ask Me	Ask Me	Ask Me : Ask Me
<input type="checkbox"/> Entire Common Name	Ask Me	Ask Me	Ask Me : Ask Me
<input type="checkbox"/> Entire Legal Identity	Ask Me	Ask Me	Ask Me : Ask Me
<input type="checkbox"/> Employment Identity	Ask Me	Ask Me	Ask Me : Ask Me
<input type="checkbox"/> Postal Addresses	Never Allow	Never Allow	Ask Me : Ask Me
<input type="checkbox"/> Contact Profiles	Ask Me	Ask Me	Ask Me : Ask Me
<input type="checkbox"/> 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			
<a href="#">Edit Policy</a> ▼			6 Item(s)
<input type="checkbox"/> Policy	Query Policy	Modify Policy	Inherited
<input type="checkbox"/> Address Type	Always Allow	Always Allow	Never Allow : Never Allow
<input type="checkbox"/> NickName	Ask Me	Ask Me	Never Allow : Never Allow
<input type="checkbox"/> Localized NickNames	Ask Me	Ask Me	Never Allow : Never Allow
<input type="checkbox"/> Comment	Ask Me	Ask Me	Never Allow : Never Allow
<input type="checkbox"/> <a href="#">Postal Address</a>	Ask Me	Ask Me	Never Allow : Never Allow
<input type="checkbox"/> Postal Addresses Extensions	Ask Me	Ask Me	Never Allow : Never Allow

The interface allows these changes 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.

## 15.2.5 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 15.2.1, “Modifying Service and Profile Details for Employee, Custom, and Personal Profiles,” on page 387.](#)
  - ♦ [Section 15.2.2, “Modifying Details for Authentication, Discovery, LDAP, and User Interaction Profiles,” on page 389.](#)

## 15.3 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 about how to configure Access Manager for secrets, see [Section 3.1.4, “Configuring a User Store for Secrets,”](#) on page 111.
- ♦ For general information about Novell SecretStore, see the [Novell SecretStore Administration Guide](http://www.novell.com/documentation/secretstore33/pdfdoc/nssadm/nssadm.pdf) (<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 [NetIQ Access Manager 4.0 SP1 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**.

The screenshot shows the 'Credential Profile' configuration page. At the top, there's a title 'Credential Profile' and a subtitle 'Edit the details about the web service.' Below this are three tabs: 'Details' (selected), 'Descriptions', and 'Custom Attribute Names'. The 'General Settings' section contains a 'Display name' field with the value 'Credential Profile' and a checkbox 'Have Discovery Encrypt This Service's Resource Ids' which is unchecked. The 'Credential Profile Settings' section has a checkbox 'Allow End Users to See Credential Profile' which is checked. The 'Local Storage of Secrets' section has a subtitle 'Access Manager controls the storage and encryption of secrets.' and contains an 'Encryption Password Hash Key' field with masked characters '\*\*\*\*\*' and a 'Preferred Encryption Method' dropdown menu set to 'Password Based Encryption With MD5 And DES'. At the bottom, there's an 'Extended Schema User Store References' section with 'New' and 'Delete' links, a count of '0 Item(s)', and a table with columns 'User Store' and 'Attribute Name'. The table is empty with the text 'No items' below it. At the very bottom are 'OK', 'Cancel', and 'Apply' buttons.

- 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 the 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, ensure 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 by 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 111.](#))

- ♦ 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.

---

**NOTE:** Do not use this LDAP attribute in Policy configuration as shared secrets. Instead you create the shared secrets attributes. The Shared secret attributes are populated in the configured LDAP attribute, and are used by policy for mapping. For more information about how to create shared secret, see ["Creating a Form Fill Policy"](#) in the *NetIQ Access Manager 4.0 SP1 Policy Guide*.

---

- ♦ 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 to add this user store reference.

5c Click **OK** twice.

6 On the Identity Server page, update the Identity Server.

## 15.4 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.

The screenshot shows the 'Informal Name' page in the Administration Console. The page has a breadcrumb trail: 'Identity Servers > TradingCo-ids > Personal Profile'. Below the breadcrumb, there is a title 'Informal Name' and a subtitle 'Create and delete custom names for the attribute Informal Name.' There are two buttons: 'New' and 'Delete'. A table with the following columns is shown: 'Custom Name' and 'Language'. The table contains one row with the values 'Juan' and 'Spanish'. A modal dialog titled 'New Custom Name' is open, prompting the user to enter a new custom name and language. The dialog has input fields for 'Custom Name' (containing 'Martin') and 'Language' (set to 'French'), and 'OK' and 'Close' buttons.

- 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.

## 15.5 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.

## 15.6 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 15.6.1, “Configuring One-to-One Attribute Maps,” on page 398.](#)
- ♦ **Employee Type:** Maps the Employee Type attribute to an LDAP attribute, then maps the possible Liberty values to LDAP values. See [Section 15.6.2, “Configuring Employee Type Attribute Maps,” on page 401.](#)
- ♦ **Employee Status:** Maps the Employee Status attribute to an LDAP attribute, then maps the possible Liberty values to LDAP values. See [Section 15.6.3, “Configuring Employee Status Attribute Maps,” on page 402.](#)
- ♦ **Postal Address:** Maps the Postal Address attribute to either multiple LDAP attributes or a delimited LDAP attribute. See [Section 15.6.4, “Configuring Postal Address Attribute Maps,” on page 404.](#)
- ♦ **Contact Method:** Maps the Contact Method attribute to multiple LDAP attributes. See [Section 15.6.5, “Configuring Contact Method Attribute Maps,” on page 405.](#)
- ♦ **Gender:** Maps the Gender attribute to an LDAP attribute, then maps the possible Liberty values to LDAP values. See [Section 15.6.6, “Configuring Gender Attribute Maps,” on page 407.](#)
- ♦ **Marital Status:** Maps the Marital Status attribute to an LDAP attribute, then maps the possible Liberty values to LDAP values. See [Section 15.6.7, “Configuring Marital Status Attribute Maps,” on page 408.](#)

**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.

## 15.6.1 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.

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.

**Default One-To-One Ldap Attribute Mapping** ?

---

**Personal Profile Single Valued Data Items to LDAP Attributes**

Data Item Name:	Ldap Attribute Name:	Access Rights:
Informal Name	<input type="text"/>	Read Only <input type="button" value="v"/>
Every Day Name	fullName	Read Only <input type="button" value="v"/>
Common Personal Title	title	Read Only <input type="button" value="v"/>
Common First Name	givenName	Read Only <input type="button" value="v"/>
Common Last Name	sn	Read Only <input type="button" value="v"/>
Common Middle Name	<input type="text"/>	Read Only <input type="button" value="v"/>
Legal Name	<input type="text"/>	Read Only <input type="button" value="v"/>
Legal Personal Title	<input type="text"/>	Read Only <input type="button" value="v"/>
Legal First Name	<input type="text"/>	Read Only <input type="button" value="v"/>
Legal Last Name	<input type="text"/>	Read Only <input type="button" value="v"/>
Legal Middle Name	<input type="text"/>	Read Only <input type="button" value="v"/>
Legal Fiscal Identification Type	<input type="text"/>	Read Only <input type="button" value="v"/>
Legal Fiscal Identification Value	<input type="text"/>	Read Only <input type="button" value="v"/>

OK Cancel

## 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.

## Default One-To-One Ldap Attribute Mapping

### Personal Profile Multiple Valued Data Items to LDAP Attributes

Data Item Name:	Ldap Attribute Name:	Access Rights:
Alternate Every Day Name	<input type="text"/>	Read Only ▾
Alternate Department Names	<input type="text"/>	Read Only ▾
Spoken or Understood Languages	<input type="text"/>	Read Only ▾

### Employee Profile Single Valued Data Items to LDAP Attributes

Data Item Name:	Ldap Attribute Name:	Access Rights:
Id	<input type="text"/>	Read Only ▾
Date of Hire	<input type="text"/>	Read Only ▾
Job Start Date	<input type="text"/>	Read Only ▾
Status	<input type="text"/>	Read Only ▾
Type	<input type="text"/>	Read Only ▾
Internal Job Title	<input type="text"/>	Read Only ▾
Department	<input type="text" value="ou"/>	Read Only ▾

OK

Cancel

## 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.



## Custom Profile Single Valued Data Items to LDAP Attributes

Data Item Name:	Ldap Attribute Name:	Access Rights:
Customizable String One	<input type="text"/>	Read Only ▾
Customizable String Two	<input type="text"/>	Read Only ▾
Customizable String Three	<input type="text"/>	Read Only ▾
Customizable String Four	<input type="text"/>	Read Only ▾
Customizable String Five	<input type="text"/>	Read Only ▾
Customizable String Six	<input type="text"/>	Read Only ▾
Customizable String Seven	<input type="text"/>	Read Only ▾
Customizable String Eight	<input type="text"/>	Read Only ▾
Customizable String Nine	<input type="text"/>	Read Only ▾
Customizable String Ten	<input type="text"/>	Read Only ▾

## Custom Profile Multiple Valued Data Items to LDAP Attributes

Data Item Name:	Ldap Attribute Name:	Access Rights:
Customizable Multi-Valued Strings One	<input type="text"/>	Read Only ▾
Customizable Multi-Valued Strings Two	<input type="text"/>	Read Only ▾

**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 15.4, "Customizing Attribute Names,"](#) on page 396). 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.

### 15.6.2 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**.

### New Employee Type LDAP Attribute Mapping

Specify name, description, user stores and mapping data.

Name:

Description:

Access Rights:

User stores:

Available user stores:

### Employee Type to LDAP Attribute

LDAP Attribute Name:

### Liberty Profile Values to LDAP Attribute Values

Employee Type Value:	LDAP Attribute Value:
Contractor Part Time:	<input type="text" value="Contractor Part Time"/>
Contractor Full Time:	<input type="text" value="Contractor Full Time"/>

- 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.

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.

- In the **LDAP Attribute Name** field, type the LDAP attribute name that you want to map to the Liberty Employee Type attribute.

- 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.

- Click **Finish**.
- On the LDAP Attribute Mapping page, click **OK**.
- Update the Identity Server.

## 15.6.3 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.

- In the Administration Console, click **Devices > Identity Servers > Edit > Liberty > LDAP Attribute Mapping > New > Employee Status**.

**New Employee Status LDAP Attribute Mapping** ?

Specify name, description, user stores and mapping data.

Name:

Description:

Access Rights:

User stores:

Available user stores:

**Employee Status to LDAP Attribute**

LDAP Attribute Name:

**Liberty Profile Values to LDAP Attribute Values**

Employee Status Value:	LDAP Attribute Value:
Active:	<input type="text" value="Active"/>
Trial:	<input type="text" value="Trial"/>
Laid Off:	<input type="text" value="Laid Off"/>
Retired:	<input type="text" value="Retired"/>
Stop Pay:	<input type="text" value="Stop Pay"/>

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.

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.

## 15.6.4 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**.

The screenshot shows a web-based configuration interface titled "New Postal Address LDAP Attribute Mapping". The interface includes the following sections:

- Name:** A text input field.
- Description:** A text input field.
- Access Rights:** A dropdown menu currently set to "Read".
- User stores:** A list box containing "<Default User Store>".
- Available user stores:** A list box containing "Installed User Store".
- Mode of Operation:** A dropdown menu currently set to "Multiple Ldap Attributes".
- Postal Address to LDAP Attribute(s):** A group of five text input fields for mapping specific attributes:
  - Postal Address Ldap Attribute:
  - Postal Code Attribute:
  - City Ldap Attribute:
  - State Ldap Attribute:
  - Country Ldap Attribute:

- 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.

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.

## 15.6.5 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**.

New Contact Method LDAP Attribute Mapping

Specify name, description, user stores and mapping data.

Name:

Description:

Access Rights

Read

User stores:

<Default User Store>

Available user stores:

Installed User Store

Contact Method to LDAP Attributes

Provider Ldap Attribute:

Account Ldap Attribute:

SubAccount Ldap Attribute:

Contact Method Template Data

Nickname:

Type:

Personal

Method:

Voice

Technology:

<Select Technology>

## 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.

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.

## 15.6.6 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**.

**New Gender LDAP Attribute Mapping** ?

Specify name, description, user stores and mapping data.

Name:

Description:

Access Rights:

User stores:  Available user stores:

**Gender to LDAP Attribute**

LDAP Attribute Name:

**Liberty Profile Values to LDAP Attribute Values**

Gender Value: LDAP Attribute Value:

Male:

Female:

- 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.

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.

## 15.6.7 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**.

Marital Status Value:	LDAP Attribute Value:
Single:	Single
Married:	Married
Common Law Marriage:	Common Law Marriage
Separated:	Separated
Divorced:	Divorced

- 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.

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.



# 16 Maintaining an Identity Server

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.

- ♦ [Section 16.1, “Managing an Identity Server,” on page 411](#)
- ♦ [Section 16.2, “Editing Server Details,” on page 413](#)
- ♦ [Section 16.3, “Configuring Component Logging,” on page 414](#)
- ♦ [Section 16.4, “Configuring Session-Based Logging,” on page 416](#)
- ♦ [Section 16.5, “Monitoring the Health of an Identity Server,” on page 422](#)
- ♦ [Section 16.6, “Monitoring Identity Server Statistics,” on page 427](#)
- ♦ [Section 16.7, “Monitoring API for the Identity Server Statistics,” on page 435](#)
- ♦ [Section 16.8, “Enabling Identity Server Audit Events,” on page 449](#)
- ♦ [Section 16.9, “Monitoring Identity Server Alerts,” on page 452](#)
- ♦ [Section 16.10, “Viewing the Command Status of the Identity Server,” on page 452](#)

## 16.1 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**.

Identity Servers

Servers		Sharable Settings					
New Cluster...   Start   Stop   Refresh   Actions▼							
<input type="checkbox"/>	Name	Status	Health	Alerts	Commands	Statistics	Configuration
<input type="checkbox"/>	<a href="#">ag42.amlab.net</a>	Current		0		<a href="#">View</a>	<a href="#">Edit</a>
<input type="checkbox"/>	<a href="#">10.10.16.61</a>	Current		0	<a href="#">Complete</a>	<a href="#">View</a>	
<input type="checkbox"/>	<a href="#">idp-51.amlab.net</a>	Current		0		<a href="#">View</a>	<a href="#">Edit</a>
<input type="checkbox"/>	<a href="#">10.10.16.51</a>	Current		0	<a href="#">Complete</a>	<a href="#">View</a>	

- 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 18](#).

**Start:** Starts the selected server. (See [Section 16.1.2, “Restarting the Identity Server,” on page 413](#).)

**Stop:** Stops the selected server. (See [Section 16.1.2, “Restarting the Identity Server,” on page 413](#).)

**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 22](#) 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 26](#) 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	<p>Lists the status of each configuration.</p> <p><b>Current:</b> Indicates that the server is using the latest configuration data. If you change a configuration, the system displays an <b>Update</b> or <b>Update All</b> link.</p> <p><b>Update:</b> A link to update an Identity Server's configuration data without stopping the server.</p> <p><b>Update All:</b> 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.</p> <p>For more information about the update process, see <a href="#">Section 16.1.1, “Updating an Identity Server Configuration,” on page 412</a>.</p>
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 <a href="#">Section 16.6, “Monitoring Identity Server Statistics,” on page 427</a> .
Configuration	Lists the Identity Server configuration to which this server belongs.

## 16.1.1 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.

## 16.1.2 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 24](#)).

- 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.

## 16.2 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 Access Manager Devices”](#) in the *NetIQ Access Manager 4.0 SP1 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**.

## 16.3 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:

- ♦ [Section 16.3.1, “Enabling Component Logging,” on page 414](#)
- ♦ [Section 16.3.2, “Managing Log File Size,” on page 416](#)

### 16.3.1 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/nam/logs/idp/tomcat/catalina.out` (Linux), 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 `/var/opt/novell/nam/logs/idp/nidplogs`.

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.

**WSTrust:** Logs events specific to the WS-Trust 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 about configuring Novell Audit Logging, see [Section 16.8, “Enabling Identity Server Audit Events,” on page 449](#).
- 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.

## 16.3.2 Managing Log File Size

On Linux, the logrotate daemon manages the log files located in the following directories:

```
/opt/novell/nam/logs
/opt/volera/roma/logs/
```

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:

```
/opt/novell/nam/idp/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-tomcat7
novell-devman
```

For information about the parameters in these files, see the documentation for the logrotate daemon.

## 16.4 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 NetIQ 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 16.4.1, “Creating the Administrator Class, Method, and Contract,” on page 417](#)
- ♦ [Section 16.4.2, “Creating the Logging Session Class, Method, and Contract,” on page 418](#)
- ♦ [Section 16.4.3, “Enabling Basic Logging,” on page 419](#)
- ♦ [Section 16.4.4, “Responding to an Incident,” on page 420](#)

## 16.4.1 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 418](#).

## 16.4.2 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 **Next**, 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 419](#).

## 16.4.3 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 about using this feature to respond to a problem, see [“Responding to an Incident” on page 420](#).

## 16.4.4 Responding to an Incident

The following sections explain how to use the feature when a user reports a problem:

- ♦ [“Creating a Logging Ticket” on page 420](#)
- ♦ [“Enabling a Logging Session” on page 421](#)
- ♦ [“Viewing the Log File” on page 422](#)

### 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 16.4.1, “Creating the Administrator Class, Method, and Contract,” on page 417](#).

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 419](#)), 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. Ensure that 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 16.4.2, “Creating the Logging Session Class, Method, and Contract,” on page 418](#)). 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

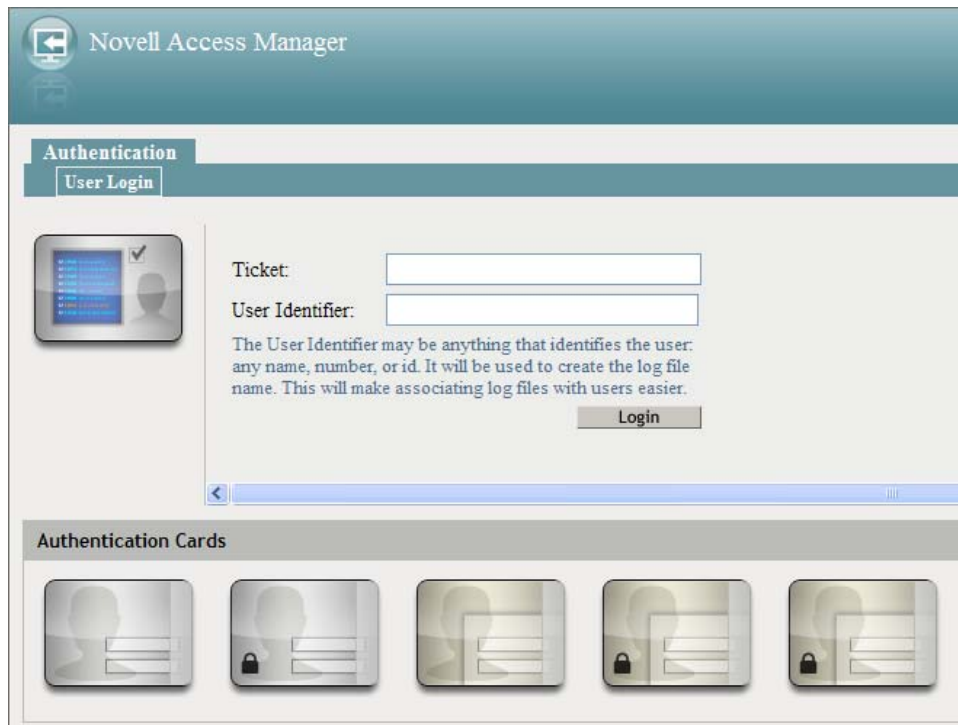
These steps are performed by the user. The URL needs to be sent to the user, with the ID and ticket values that were specified in [“Creating a Logging Ticket” on page 420](#).

- 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/nam/logs/idp/nidplogs`

**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:** `/opt/novell/nam/idp/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.**

## 16.5 Monitoring the Health of an Identity Server

- ♦ [Section 16.5.1, "Health States," on page 423](#)
- ♦ [Section 16.5.2, "Viewing the Health Details of an Identity Server," on page 423](#)
- ♦ [Section 16.5.3, "Viewing the Health Details of a Cluster," on page 426](#)

## 16.5.1 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 is not running or functional, or that the server is having a runtime problem.


## 16.5.2 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**.

GeneralHealthAlertsCommand StatusStatistics

Refresh | Update from ServerLast Reported Time: September 24,

Status	Description
	Server is operational (Passed)

Services Detail

Type	Status	Message
Services		Identity Server Configuration Configuration Datastore User Datastores Signing and Encryption Keys
Identity Server Configuration		Fully applied
Configuration Datastore		Operating properly
User Datastores		Operating properly
Signing and Encryption Keys		Signing key available Encryption key available

Close

The status icon is followed by a description that explains the significance of the current state. For more information about the icons, see [Section 16.5.1, “Health States,” on page 423](#).

- 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
<b>Status:</b> Indicates whether the Identity Server is online and operational.	<p>Verify whether the Identity Server has been stopped or is not configured.</p> <p>Also verify that network problems are not interfering with communications between the Identity Server and the Administration Console.</p>
<b>Services:</b> Indicates the general health of all configured services.	<p>If one service is unhealthy, this category reflects that status. See the particular service that also displays an unhealthy status.</p>
<b>Identity Server Configuration:</b> Indicates the status of the configuration.	<p>Configure the Identity Server or assign the server to a configuration. See <a href="#">Chapter 1, “Configuring an Identity Server,”</a> on page 17.</p>
<b>Configuration Datastore:</b> Indicates the status of the installed configuration datastore.	<p>You might need to restart Tomcat or reinstall the Administration Console.</p> <p>If you have a backup Administration Console, you can restore it. See <a href="#">“Backing Up and Restoring”</a> in the <i>NetIQ Access Manager 4.0 SP1 Administration Console Guide</i>.</p> <p>If you don’t have a backup, you can try repairing the configuration datastore. See <a href="#">“Repairing the Configuration Datastore”</a> in the <i>NetIQ Access Manager 4.0 SP1 Administration Console Guide</i>.</p> <p>If you want to convert a secondary console to your primary console, see <a href="#">“Converting a Secondary Administration Console into a Primary Console”</a> in the <i>NetIQ Access Manager 4.0 SP1 Administration Console Guide</i>.</p>
<b>User Datastores:</b> Indicates whether the Identity Server can communicate with the user stores, authenticate as the admin user, and find the search context.	<p>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 <a href="#">Section 3.1, “Configuring Identity User Stores,”</a> on page 106.</p>
<b>Signing, Encryption and SSL Connector Keys:</b> Indicates whether these keystores contain valid a key.	<p>Click <b>Identity Servers &gt; Edit &gt; Security</b> and replace any missing or expired keys.</p>
<b>System Incoming and Outgoing HTTP Requests:</b> 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.	<p>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.</p> <p>If a cluster member is slow because it doesn’t have enough physical resources (speed or memory) to handle the load, upgrade the hardware.</p>
<b>SSL Communication:</b> Indicates whether SSL communication is operating correctly. This health check appears only when the SSL communication check fails.	<p>Check SSL connectivity. Check for expired SSL certificates.</p>

Status Category	If not healthy
<b>Audit Logging Server:</b> 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 <b>Devices &gt; Identity Servers &gt; Edit &gt; Logging</b> ).	Check the network connection between the Identity Server and the auditing server.  See "Troubleshooting Novell Audit" ( <a href="http://www.novell.com/documentation/novellaudit20/novellaudit20/data/al0lh30.html">http://www.novell.com/documentation/novellaudit20/novellaudit20/data/al0lh30.html</a> ).

- 4 Click **Close**.

## 16.5.3 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 screenshot shows the 'Health' tab of the Administration Console. At the top, there are tabs for 'General', 'Health', 'Alerts', 'Command Status', and 'Statistics'. Below the tabs, there are links for 'Refresh' and 'Update from Server', and a timestamp 'Last Reported Time: September 24, 2011'. The main content area shows a status icon (a green circle with a white checkmark) and the text 'Server is operational (Passed)'. Below this is a 'Services Detail' table with columns 'Type', 'Status', and 'Message'.

Type	Status	Message
Services		Identity Server Configuration Configuration Datastore User Datastores Signing and Encryption Keys
Identity Server Configuration		Fully applied
Configuration Datastore		Operating properly
User Datastores		Operating properly
Signing and Encryption Keys		Signing key available Encryption key available

At the bottom of the page, there is a 'Close' button.

The status icon is followed by a description that explains the significance of the current state. For more information about the icons, see [Section 16.5.1, "Health States," on page 423](#).

- 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.

## 16.6 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**.

General

Health

Alerts




Command Status

Statistics

Server Activity

[ Statistics | [Live Statistics Monitoring](#) ]

Server Activity

Application		
Free Memory	86.11 %	 <a href="#">Graphs</a>
Authentications		
Provided Authentications	5	
Consumed Authentications	3	
Provided Authentications Failures	0	
Consumed Authentications Failures	0	
Logouts	3	
Cached Sessions	0	 <a href="#">Graphs</a>
Cached Ancestral Sessions	0	
Cached Subjects	0	
Cached Principals	0	
Cached Artifacts	0	
Incoming HTTP Requests		
Total Requests	1152	 <a href="#">Graphs</a>
Currently Active Requests	0	
Oldest Active Request (Milliseconds)	0	
Last Interval Maximum Request Duration (Milliseconds)	0	
Last Interval Mean Request Duration (Milliseconds)	0	
Historical Maximum Request Duration (Milliseconds)	1070	
Historical Mean Request Duration (Milliseconds)	4	

- 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](#)
- ♦ [SP Brokering](#)

5 Click **Close** to return to the Servers page.

---

**NOTE:** The statistics graphs of the Identity Server and Access Gateway are available in only the primary Administration Console. The periodic stats are sent to the secondary Administration Console only when the primary console is not available. Hence, the statistics graphs of the Identity Server and Access Gateway do not display any statistics values in the secondary Administration Console.

---

## 16.6.1 Application

Statistic	Description
Free Memory	The percentage of free memory available to the JVM (Java Virtual Machine). Click <b>Graphs</b> 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.

## 16.6.2 Authentications

Statistic	Description
Provided Authentications	The number of successful provided authentications given out to external entities after the Identity Server was started.
Consumed Authentications	The number of successful consumed authentications after the Identity Server was started.
Provided Authentication Failures	The number of failed provided authentications given out to external entities after the Identity Server was started.
Consumed Authentication Failures	The number of failed consumed authentications after the Identity Server was started.
Historical Maximum Logins Served	The maximum number of logins served during an interval and displayed after completion of the interval.
Logins In Last Interval	The number of active user sessions during the last interval.
Logouts	The number of explicit logouts performed by users. This does not include logouts where an inactive session was destroyed.

Statistic	Description
Cached Sessions	<p>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.</p> <p>Click <b>Graphs</b> 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.</p>
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.
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.

### 16.6.3 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 after the Identity Server was started. Click <b>Graphs</b> 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.

Statistic	Description
Historical Maximum Request Duration (Milliseconds)	The age of the longest incoming HTTP request that was processed after the Identity Server was started.
Historical Mean Request Duration (Milliseconds)	The mean age of all incoming HTTP requests that were processed after the Identity Server was started.

## 16.6.4 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 after the Identity Server was started. Click <b>Graphs</b> 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 after the Identity Server was started.
Historical Mean Request Duration (Milliseconds)	The mean age of all outgoing HTTP requests that were processed after the Identity Server was started.

## 16.6.5 Liberty

Statistic	Description
Liberty Federation	The number of Liberty protocol federations performed after the Identity Server was started.
Liberty De-Federations	The number of Liberty protocol defederations performed after the Identity Server was started.
Liberty Register-Names	The number of Liberty protocol register names performed after the Identity Server was started.

## 16.6.6 SAML 1.1

Statistic	Description
SAML1.1 Attribute Queries	The number of SAML 1.1 protocol attribute queries performed after the Identity Server was started.

## 16.6.7 SAML 2

Statistic	Description
SAML2 Attribute Queries	The number of SAML 2 protocol attribute queries performed after the Identity Server was started.
SAML2 Federations	The number of SAML 2 protocol federations performed after the Identity Server was started.
SAML2 Defederations	The number of SAML 2 protocol defederations performed after the Identity Server was started.
SAML2 Register-Names	The number of SAML 2 protocol register names performed after the Identity Server was started.

## 16.6.8 WSF (Web Services Framework)

Statistic	Description
Personal Profile Service Queries	The number of Liberty IDSIS Personal Profile Web Service queries performed after the Identity Server was started.
Personal Profile Service Modifies	The number of Liberty IDSIS Personal Profile Web Service changes performed after the Identity Server was started.
Employee Profile Service Queries	The number of Liberty IDSIS Employee Profile Web Service queries performed after the Identity Server was started.
Employee Profile Service Modifies	The number of Liberty IDSIS Employee Profile Web Service changes performed after the Identity Server was started.
Custom Profile Service Queries	The number of Novell Custom Profile Web Service queries performed after the Identity Server was started.
Custom Profile Service Modifies	The number of Novell Custom Profile Web Service changes performed after the Identity Server was started.
Credential Profile Service Queries	The number of Novell Credential Profile Web Service queries performed after the Identity Server was started.
Credential Profile Service Modifies	The number of Novell Credential Profile Web Service changes performed after the Identity Server was started.
Authentication Profile Service Queries	The number of Novell Authentication Profile Web Service queries performed after the Identity Server was started.

<b>Statistic</b>	<b>Description</b>
Authentication Profile Service Modifies	The number of Novell Authentication Profile Web Service changes performed after the Identity Server was started.
LDAP Profile Service Queries	The number of Novell LDAP Profile Web Service queries performed after the Identity Server was started.
LDAP Profile Service Modifies	The number of Novell LDAP Profile Web Service changes performed after the Identity Server was started.
Constant Profile Service Queries	The number of Novell Constant Profile Web Service queries performed after the Identity Server was started.
Discovery Service Queries	The number of Liberty Discovery Web Service queries performed after the Identity Server was started.
Discovery Service Modifies	The number of Liberty Discovery Web Service changes performed after the Identity Server was started.
Redirected Interaction Service Requests	The number of Liberty User Interaction Redirection Profile requests performed after the Identity Server was started.
Trusted Interaction Service Requests	The number of Liberty User Interaction Trusted Service Profile requests performed after the Identity Server was started.
Client of Redirected Interaction Service Requests	The number of Liberty User Interaction Redirection Profile requests initiated as a client after 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 after 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 after 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 after 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 after 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 after 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 after 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 after 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 after the Identity Server was started.



Statistic	Description
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 after 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.

## 16.6.9 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. Ensure 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 after 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 after 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 after the Identity Server was started.
Authoritative Server Obtained from Cookie	The total number of authoritative servers identified by using the HTTP cookie after the Identity Server was started.
Payload Examinations	The total number of attempted payload examinations to identify the authoritative server after the Identity Server was started.

<b>Statistic</b>	<b>Description</b>
Successful Payload Examinations	The total number of successful payload examinations to identify the authoritative server after the Identity Server was started.
Identity ID Broadcasts	The total number of attempted Identity ID Broadcasts to identify the authoritative server after the Identity Server was started.
Successful Identity ID Broadcasts	The total number of successful Identity ID Broadcasts to identify the authoritative server after 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 after the Identity Server was started.

## 16.6.10 LDAP

<b>Statistic</b>	<b>Description</b>
User Store Replica Restarts	The number of times that a user store replica became unavailable so that a restart was necessary after 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 after 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 after the Identity Server was started.
Currently Active Connection Waits	The current number of user threads waiting for an LDAP connection to become available.
Connection Waits	The number of times that a user thread was required to wait for an LDAP connection to become available after 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 after 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 after 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.

## 16.6.11 SP Brokering

<b>Statistic</b>	<b>Description</b>
Total Brokering Requests	The total number of brokering requests created after 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.

Statistic	Description
Total Brokering Requests Denied Due to Group Check	The total number of brokering authentication requests denied in a target service provider. The brokering group can either be the identity provider or target service provider but both does not belong to the same group.
Total Brokering Requests Denied Due to Role Deny	The total number of brokering authentication requests to a target service provider denied due to broker policy evaluation denying the role.
Total Brokering Requests Passed	The total number of brokering requests passed after the Identity Server was started.

## 16.7 Monitoring API for the Identity Server Statistics

For programmatic access to the Identity Server statistics, you must enable the Representational State Transfer (REST) API.

To enable the REST API:

- 1 Place the `nidpmonitor.txt` file in to the `WEB-INF` directory of the Identity Server and ESP webapp.

For Identity Server:

**Linux:** `/opt/novell/nam/idp/webapps/nidp/WEB-INF/`

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

For ESP:

**Linux:** `/opt/novell/nam/mag/webapps/nesp/WEB-INF/`

**Windows:** `\Program Files\Novell\Tomcat\webapps\nesp\WEB-INF`

- 2 Add the following line in `nidpmonitor.txt`:

```
urn:novell:nidp:monitor:anyaccess
```

After this line, you must add the IP addresses of the servers from which you will be making calls to the REST API. Example content of the `nidpmonitor.txt` file:

```
urn:novell:nidp:monitor:anyaccess
```

```
10.0.0.0
```

```
172.16.0.0
```

- 3 Restart the Identity Server.

---

**IMPORTANT:** Frequent requests to get the statistics impact the system performance. It is recommended to keep a five minutes interval between every probe for the statistics.

---

### 16.7.1 Endpoints of the REST API

The Identity Server uses this REST endpoint: `https://<DNS FQDN of NIDP>:<port>/nidp/app/monitor`.

ESP uses this REST endpoint: `https://<DNS FQDN of ESP>:<port>/nesp/app/monitor`.

The endpoint takes the following three parameters:

Parameter	Value	Description
displayType	XML	This parameter specifies the output display type. Currently it supports only XML.
command	See <a href="#">Supported Commands and Their Outputs</a> for details of the commands which support this parameter.	This specifies the monitored statistics that are to be displayed.
reset	This parameter can take only "True" as value. See <a href="#">Supported Commands and Their Outputs</a> for details of the commands which support reset.	This specifies the monitored statistics that is to be reset.

## 16.7.2 Supported Commands and Their Outputs

The following list includes supported commands:

- ♦ [httpInRequests](#)
- ♦ [inUrlTypes](#)
- ♦ [httpOutRequests](#)
- ♦ [ldapServerConfig](#)
- ♦ [ldapConnections](#)
- ♦ [ldapConnectionWaits](#)
- ♦ [ldapReplicaStats](#)
- ♦ [ldapPerfOverview](#)
- ♦ [ldapFailOverview](#)
- ♦ [authPerf](#)

**NOTE:** When using the curl command, place the URL inside double quotes (""). Otherwise, the XML data does not render. For example, curl -k "https://<domain>:<port>/nidp/app/monitor?command=inUrlTypes&displayType=xml".

### httpInRequests

This command supports reset. This command displays the monitored statistics of incoming HTTP requests to the Identity Server.

Example output:

```
<?xml version="1.0" encoding="UTF-8"?>
<InComingHTTPRequests>
  <ThreadIntervals>
    <NamedValues>
      <NamedValue name="Total" value="61" />
      <NamedValue name="Current Requests" value="1" />
    
```

```

</NamedValues>
<ActiveObjects abandoned="0">
    <ActiveObject name="ajp-bio-/127.0.0.1-9019-exec-23" age="3">
    </ActiveObject>
</ActiveObjects>
<Historical>
    <Spectrometer dataPoints="22" totalCount="60"
        maxDataPoints="500">
        <max>145</max>
        <min>1</min>
        <mean>18</mean>
    </Spectrometer>
</Historical>
</ThreadIntervals>
</InComingHTTPRequests>

```

## inUrlTypes

This command supports reset. This command displays counts of the URL types and services that have been requested to the Identity Server.

Example output:

```

<UrlTypes>
    <NamedValues>
        <NamedValue name="CMD: /app/, monitor" value="15" />
        <NamedValue name="CMD: /app/, ping" value="13" />
        <NamedValue name="CMD: /idff, soap" value="1" />
        <NamedValue name="CMD: /idff, sso" value="4" />
        <NamedValue name="JSP: content.jsp" value="1" />
    </NamedValues>
</UrlTypes>

```

## httpOutRequests

This command supports reset. This command displays the monitored statistics of outgoing HTTP requests from the Identity Server.

Example output:

```

<?xml version="1.0" encoding="UTF-8"?>
<OutGoingHTTPRequests>

```

```

<ThreadIntervals>
  <NamedValues>
    <NamedValue name="Total" value="25" />
  </NamedValues>
</Historical>
  <Spectrometer dataPoints="10" totalCount="25" maxDataPoints="500">
    <max>51</max>
    <min>2</min>
    <mean>12</mean>
  </Spectrometer>
</Historical>
</ThreadIntervals>
</OutGoingHTTPRequests>

```

## IdapServerConfig

This command does not support reset. This command displays the setup details of the Identity Server configuration store and the user store.

Example output:

```

<UserStoreManager id="MGf373f25e-5a95-484e-85fe-2d3f073e3c28">
  <TrustConfigDataStore>
    <UserStore id="USef25d609-7577-4bab-a705-f00b5406f2cc"

systemId="cn=SCC7u0ouw,cn=cluster,cn=nids,ou=accessManagerContainer,o=novell"

    displayName="" directoryName="Novell eDirectory"

adminUserName="ou=nidsUser,ou=UsersContainer,ou=Partition,ou=PartitionsContainer,o
u=VCDN_Root,ou=accessManagerContainer,o=novell"

    idleTimeout="10000" bindTimeout="0" allowRebind="true" maxWaitReservations="-
1">

    <Replicas>
      <Replica id="0c498978-2d16-4b25-ae41-484fca62fc36"
systemId="PseudoXMLBasedUserStoreReplicaDN0"

        displayName="Replica 1" host="ldaps:// 10.0.0.0" port="636"

        maxConnections="5" doSSL="true">

        <ConnectionPool

          id="PL8928e311-6a84-494a-b61a-5ff43005dd6f:0c498978-2d16-4b25-ae41-
484fca62fc36"

```

```
adminUserName="ou=nidsUser,ou=UsersContainer,ou=Partition,ou=PartitionsContainer,o
u=VCDN_Root,ou=accessManagerContainer,o=novell"
```

```
maxConnections="5" skipCount="10" waitResTimeout="60000"
```

```
waitResSleep="20" waitResSleepIterCount="3000" load="0">
```

```
<AdminConnections>
```

```
<Connection id="0adff495-9321-485c-b156-66deceeeefa84"
```

```
type="admin" checkedOut="false" IdleAge="5985087" />
```

```
</AdminConnections>
```

```
</ConnectionPool>
```

```
</Replica>
```

```
</Replicas>
```

```
</UserStore>
```

```
</TrustConfigDataStore>
```

```
<UserStores>
```

```
<UserStore id="USc15e7906-d4a9-41c3-8438-cd10fb6c7a89"
```

```
systemId="cn=USmkp9m,cn=Alrre4,cn=SCC7u0ouw,cn=cluster,cn=nids,ou=accessManagerCon
tainer,o=novell"
```

```
displayName="SingleBoxUserStore" directoryName="Novell eDirectory"
```

```
adminUserName="cn=admin,o=novell" idleTimeout="10000" bindTimeout="0"
```

```
allowRebind="true" maxWaitReservations="-1">
```

```
<SearchContexts>
```

```
<SearchContext order="0" scope="1" context="o=novell" />
```

```
</SearchContexts>
```

```
<Replicas>
```

```
<Replica id="0a307605-8946-4455-8080-f1819562481d"
```

```
systemId="cn=USRlxx69,cn=USmkp9m,cn=Alrre4,cn=SCC7u0ouw,cn=cluster,cn=nids,ou=acc
essManagerContainer,o=novell"
```

```
displayName="SingleBoxUserStoreReplica" host="ldaps:// 10.0.0.0"
```

```
port="636" maxConnections="20" doSSL="true">
```

```
<ConnectionPool
```

```
id="PLce0653bc-488d-4e7c-81a5-08e935d83c82:0a307605-8946-4455-8080-
f1819562481d"
```

```
adminUserName="cn=admin,o=novell" maxConnections="20" skipCount="10"
```

```
waitResTimeout="60000" waitResSleep="20" waitResSleepIterCount="3000"
load="0">
```

```
<AdminConnections>
```

```

        <Connection id="b1c0a413-2c36-4b64-831c-b0849421c7a0"
            type="admin" checkedOut="false" IdleAge="259357" />
    </AdminConnections>
</ConnectionPool>
</Replica>
</Replicas>
</UserStore>
</UserStores>
</UserStoreManager>

```

## IdapConnections

This command does not support reset. This command displays counts of the Identity Server LDAP connection.

Example output:

```

<LdapConnections>
    <TotalAdded admin="25" user="1" />
    <TotalRemoved admin="23" user="1" />
    <CurrentValidInUse admin="0" user="0" />
    <CurrentValidOutOfUse admin="2" user="0" />
    <CurrentInvalidEstd admin="0" user="0" />
    <CurrentInvalidNonEstd admin="0" user="0" />
    <TotalForceCloseSuccess admin="23" user="1" />
    <TotalForceCloseError admin="0" user="0" />
    <TotalForceCloseNonEstd admin="0" user="0" />
</LdapConnections>

```

## IdapConnectionWaits

This command supports reset. This command displays statistics of the Identity Server LDAP connection wait time.

Example output:

```

<LDAPConnectionWaits>
</LDAPConnectionWaits>

```

## IdapReplicaStats

This command does not support reset. This command displays statistics of the Identity Server LDAP replica.

Example output:



```

<LdapReplicaStatsCollection>
  <TrustConfigDataStoreStats>
    <LdapReplicaStats displayName="Replica 1" host="ldaps:// 10.0.0.0 "
      inRestart="false" load="0">
      <ExistingAdminConnectionReservation admin="97" />
      <NewConnections admin="2" user="0" />
      <Rebinds user="0" />
      <InvalidRebinds user="0" />
      <Waits admin="0" user="0" />
      <WaitExpired admin="0" user="0" />
      <WaitSkipped admin="0" user="0" />
      <WaitHitMaxSkipped admin="0" user="0" />
    </LdapReplicaStats>
  </TrustConfigDataStoreStats>
  <LdapReplicaStats displayName="SingleBoxUserStoreReplica"
    host="ldaps://10.0.0.0" inRestart="false" load="0">
    <ExistingAdminConnectionReservation admin="86" />
    <NewConnections admin="28" user="1" />
    <Rebinds user="0" />
    <InvalidRebinds user="0" />
    <Waits admin="0" user="0" />
    <WaitExpired admin="0" user="0" />
    <WaitSkipped admin="0" user="0" />
    <WaitHitMaxSkipped admin="0" user="0" />
  </LdapReplicaStats>
</LdapReplicaStatsCollection>

```

## IdapPerfOverview

This command does not support reset. This command displays performance statistics of the Identity Server LDAP replica.

Example output:

```

<?xml version="1.0" encoding="UTF-8"?>
<LdapReplicaPerfCollection>
  <TrustConfigDataStorePerf>
    <LdapReplicaPerf displayName="Replica 1" inRestart="false"
      load="0" host="ldaps://10.0.0.0">

```

```

<AllOpsDuration>
  <Interval>
    <Spectrometer dataPoints="5" totalCount="6" maxDataPoints="300">
      <max>46</max>
      <min>1</min>
      <mean>16</mean>
    </Spectrometer>
  </Interval>
<Historical>
  <Spectrometer dataPoints="11" totalCount="100" maxDataPoints="500">
    <max>93</max>
    <min>1</min>
    <mean>3</mean>
  </Spectrometer>
</Historical>
</AllOpsDuration>
<CreateConnDuration>
  <Interval>
    <Spectrometer dataPoints="2" totalCount="2" maxDataPoints="300">
      <max>46</max>
      <min>44</min>
      <mean>45</mean>
    </Spectrometer>
  </Interval>
<Historical>
  <Spectrometer dataPoints="1" totalCount="1" maxDataPoints="500">
    <max>93</max>
    <min>93</min>
    <mean>93</mean>
  </Spectrometer>
</Historical>
</CreateConnDuration>
<CloseConnDuration>
  <Interval>
    <Spectrometer dataPoints="1" totalCount="2" maxDataPoints="300">
      <max>1</max>

```

```

        <min>1</min>
        <mean>1</mean>
    </Spectrometer>
</Interval>
</CloseConnDuration>
<SearchDuration>
    <Interval>
        <Spectrometer dataPoints="2" totalCount="2" maxDataPoints="300">
            <max>3</max>
            <min>2</min>
            <mean>2</mean>
        </Spectrometer>
    </Interval>
    <Historical>
        <Spectrometer dataPoints="8" totalCount="95" maxDataPoints="500">
            <max>11</max>
            <min>1</min>
            <mean>2</mean>
        </Spectrometer>
    </Historical>
</SearchDuration>
<GetDuration>
    <Historical>
        <Spectrometer dataPoints="4" totalCount="4" maxDataPoints="500">
            <max>10</max>
            <min>1</min>
            <mean>6</mean>
        </Spectrometer>
    </Historical>
</GetDuration>
<ModifyDuration></ModifyDuration>
<CreateObjDuration></CreateObjDuration>
<DeleteObjDuration></DeleteObjDuration>
<ExtDuration></ExtDuration>
<RebindDuration></RebindDuration>
</LdapReplicaPerf>

```

```

</TrustConfigDataStorePerf>
<LdapReplicaPerf displayName="SingleBoxUserStoreReplica"
  inRestart="false" load="0" host="ldaps://10.0.0.0">
  <AllOpsDuration>
    <Interval>
      <Spectrometer dataPoints="5" totalCount="19" maxDataPoints="300">
        <max>46</max>
        <min>1</min>
        <mean>13</mean>
      </Spectrometer>
    </Interval>
    <Historical>
      <Spectrometer dataPoints="5" totalCount="9" maxDataPoints="500">
        <max>43</max>
        <min>0</min>
        <mean>5</mean>
      </Spectrometer>
    </Historical>
  </AllOpsDuration>
  <CreateConnDuration>
    <Interval>
      <Spectrometer dataPoints="2" totalCount="5" maxDataPoints="300">
        <max>46</max>
        <min>45</min>
        <mean>45</mean>
      </Spectrometer>
    </Interval>
    <Historical>
      <Spectrometer dataPoints="1" totalCount="1" maxDataPoints="500">
        <max>43</max>
        <min>43</min>
        <mean>43</mean>
      </Spectrometer>
    </Historical>
  </CreateConnDuration>
  <CloseConnDuration>

```

```

<Interval>
  <Spectrometer dataPoints="1" totalCount="5" maxDataPoints="300">
    <max>1</max>
    <min>1</min>
    <mean>1</mean>
  </Spectrometer>
</Interval>
</CloseConnDuration>
<SearchDuration>
  <Interval>
    <Spectrometer dataPoints="2" totalCount="3" maxDataPoints="300">
      <max>2</max>
      <min>1</min>
      <mean>1</mean>
    </Spectrometer>
  </Interval>
</SearchDuration>
<GetDuration>
  <Interval>
    <Spectrometer dataPoints="2" totalCount="3" maxDataPoints="300">
      <max>2</max>
      <min>1</min>
      <mean>1</mean>
    </Spectrometer>
  </Interval>
<Historical>
  <Spectrometer dataPoints="2" totalCount="4" maxDataPoints="500">
    <max>2</max>
    <min>1</min>
    <mean>1</mean>
  </Spectrometer>
</Historical>
</GetDuration>
<ModifyDuration></ModifyDuration>
<CreateObjDuration></CreateObjDuration>
<DeleteObjDuration></DeleteObjDuration>

```

```

<ExtDuration>
  <Interval>
    <Spectrometer dataPoints="2" totalCount="2" maxDataPoints="300">
      <max>2</max>
      <min>1</min>
      <mean>1</mean>
    </Spectrometer>
  </Interval>
<Historical>
  <Spectrometer dataPoints="3" totalCount="4" maxDataPoints="500">
    <max>3</max>
    <min>0</min>
    <mean>1</mean>
  </Spectrometer>
</Historical>
</ExtDuration>
<RebindDuration>
  <Interval>
    <Spectrometer dataPoints="1" totalCount="1" maxDataPoints="300">
      <max>3</max>
      <min>3</min>
      <mean>3</mean>
    </Spectrometer>
  </Interval>
</RebindDuration>
</LdapReplicaPerf>
</LdapReplicaPerfCollection>

```

## IdapFailOverview

This command does not support reset. This command displays statistics of the Identity Server LDAP replica failure.

Example output:

```

<?xml version="1.0" encoding="UTF-8"?>
<LdapReplicaFailureCollection>
  <TrustConfigDataStoreFailure>
    <LdapReplicaFailurePerf displayName="Replica 1"

```

```

inRestart="false" load="0" host="ldaps://10.0.0.0">
<AllOpsDuration>
  <Historical>
    <Spectrometer dataPoints="2" totalCount="3" maxDataPoints="500">
      <max>2</max>
      <min>1</min>
      <mean>1</mean>
    </Spectrometer>
  </Historical>
</AllOpsDuration>
<CreateConnDuration></CreateConnDuration>
<CloseConnDuration></CloseConnDuration>
<SearchDuration></SearchDuration>
<GetDuration>
  <Historical>
    <Spectrometer dataPoints="2" totalCount="3" maxDataPoints="500">
      <max>2</max>
      <min>1</min>
      <mean>1</mean>
    </Spectrometer>
  </Historical>
</GetDuration>
<ModifyDuration></ModifyDuration>
<CreateObjDuration></CreateObjDuration>
<DeleteObjDuration></DeleteObjDuration>
<ExtDuration></ExtDuration>
<RebindDuration></RebindDuration>
</LdapReplicaFailurePerf>
</TrustConfigDataStoreFailure>
<LdapReplicaFailurePerf displayName="SingleBoxUserStoreReplica"
inRestart="false" load="0" host="ldaps://10.0.0.0">
<AllOpsDuration>
  <Interval>
    <Spectrometer dataPoints="2" totalCount="2" maxDataPoints="300">
      <max>3054</max>
      <min>3051</min>

```

```

        <mean>3052</mean>
      </Spectrometer>
    </Interval>
  </AllOpsDuration>
  <CreateConnDuration>
    <Interval>
      <Spectrometer dataPoints="2" totalCount="2" maxDataPoints="300">
        <max>3054</max>
        <min>3051</min>
        <mean>3052</mean>
      </Spectrometer>
    </Interval>
  </CreateConnDuration>
  <CloseConnDuration></CloseConnDuration>
  <SearchDuration></SearchDuration>
  <GetDuration></GetDuration>
  <ModifyDuration></ModifyDuration>
  <CreateObjDuration></CreateObjDuration>
  <DeleteObjDuration></DeleteObjDuration>
  <ExtDuration></ExtDuration>
  <RebindDuration></RebindDuration>
</LdapReplicaFailurePerf>
</LdapReplicaFailureCollection>

```

## authPerf

This command does not support reset. This command displays performance statistics of the Identity Server local authentication.

Example output:

```

<?xml version="1.0" encoding="UTF-8"?>
<AuthenticationPerformance>
  <NamedValues>
    <NamedValue name="Provided Authentications" value="2" />
    <NamedValue name="Consumed Authentications" value="3" />
    <NamedValue name="Consumed Authentications Failures" value="6" />
    <NamedValue name="Historical PEAK Logins" value="1" />
    <NamedValue name="Logouts" value="2" />
  </NamedValues>
</AuthenticationPerformance>

```



```

</NamedValues>
<LocalAuthDuration historicalMean="106" intervalMean="105">
  <ContractStats name="Name/Password - Form">
    <Historical>
      <Spectrometer dataPoints="1" totalCount="1" maxDataPoints="500">
        <max>100</max>
        <min>100</min>
        <mean>100</mean>
      </Spectrometer>
    </Historical>
  </ContractStats>
  <ContractStats name="MyTwoContracts">
    <Interval>
      <Spectrometer dataPoints="1" totalCount="1" maxDataPoints="300">
        <max>105</max>
        <min>105</min>
        <mean>105</mean>
      </Spectrometer>
    </Interval>
    <Historical>
      <Spectrometer dataPoints="1" totalCount="1" maxDataPoints="500">
        <max>113</max>
        <min>113</min>
        <mean>113</mean>
      </Spectrometer>
    </Historical>
  </ContractStats>
</LocalAuthDuration>
</AuthenticationPerformance>

```

## 16.8 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 *NetIQ Access Manager 4.0 SP1 Administration Console Guide*.

- 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.
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.

Event	Description
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.
Brokering Across Groups Denied	Brokering authentication request denied to a target service provider. The brokering group consists of either the Identity Provider or target Service Provider, but both does not belong to the same group.
Brokering Rule Evaluated to Deny	Brokering authentication request denied to a target service provider due to broker policy evaluation resulted in denying.
Brokering Handled	The total number of brokering authentication requests handled by the Identity Server when it started.
WebService Request Authenticated	Generated when a user is authenticated for requesting a token for a Web service.
WebService Request Authentication Failed	Generated when a user's authentication fails for requesting a token for a Web service.
Token Was Issued To WebService	Generated when a token is issued for accessing a Web service.
Token Issue To WebService Failed	Generated when a request to issue a token for accessing a Web service fails.
Token Was Validated To A WebService	Generated when a token is validated for a Web service.
Token Validation To WebService Failed	Generated when a token validation for accessing a web service fails.
Token Renewed	Generated when a token is renewed for a Web service.
Token Renew Failed	Generated when renewing a token for a Web service fails.

4 Click **Apply**, then **OK**.

5 Click **Servers > Update Servers**.

Restart the Novell Audit server.

**NOTE:** Identity server logs the IP address of client machine, from where authentication requests originate, into the audit events. If the client machine is behind proxy, then proxy IP address will be logged. If you need to skip the proxy IP address and log the actual client machine IP address, you have to configure the RemoteIP Valve in the Tomcat configuration file (`server.xml`) on all the identity servers.

The `server.xml` file can be found at `/opt/novell/nam/idp/conf/server.xml` (for linux) and `//Program File x(86)/Novell/Tomcat/conf/server.xml` (for Windows).

The details of configuring RemoteIPValve can be found at [http://tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Remote\\_IP\\_Valve](http://tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Remote_IP_Valve)

To configure audit events to capture the source IP address of the X-forwarded-header, in the `server.xml` file add the following details after the `Engine` element:

```
<Engine defaultHost="localhost" name="Catalina">  
<Valve className="org.apache.catalina.valves.RemoteIpValve"  
internalProxies="IP addresses" />
```

Substitute IP addresses with the IP address of the proxy and load balancer.

Restart Tomcat by using `rcnovell-idp restart`.

---

## 16.9 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**.

## 16.10 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 16.10.1, “Viewing the Status of Current Commands,” on page 453](#)
- [Section 16.10.2, “Viewing Detailed Command Information,” on page 453](#)

## 16.10.1 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
<b>Name</b>	Lists the Identity Server name.
<b>Status</b>	Lists the status of each server.
<b>Type</b>	Displays type of command issued to the server.
<b>Admin</b>	Displays the credentials of the administrator who performed the command.
<b>Date &amp; Time</b>	The date and time that the command was issued. Date and time entries are specified in the local time.

## 16.10.2 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.



---

# 17 Troubleshooting Identity Server and Authentication

This section provides information about the following topics:

- ♦ [Section 17.1, “Useful Networking Tools for Linux Identity Server,” on page 456](#)
- ♦ [Section 17.2, “Troubleshooting 100101043 and 100101044 Liberty Metadata Load Errors,” on page 456](#)
- ♦ [Section 17.3, “Authentication Issues,” on page 464](#)
- ♦ [Section 17.4, “Problems Reading Keystores after Identity Server Re-installation,” on page 466](#)
- ♦ [Section 17.5, “After Setting Up the User Store to Use SecretStore, Users Report 500 Errors,” on page 467](#)
- ♦ [Section 17.6, “When Multiple Browser Logout Option Is Enabled User Is Not Getting Logged Out From Different Sessions,” on page 467](#)
- ♦ [Section 17.7, “302 Redirect to 'RelayState' URL after consuming a SAML Response is being sent to an incorrect URL,” on page 467](#)
- ♦ [Section 17.8, “Configuring SAML 1.1 Identity Provider Without Specifying Port in the Login URL Field,” on page 468](#)
- ♦ [Section 17.9, “Attributes are Not Available Through Form Fill When OIOSAML Is Enabled,” on page 468](#)
- ♦ [Section 17.10, “Issue in Importing Metadata While Configuring Identity Provider or Service Provider Using Metadata URL,” on page 468](#)
- ♦ [Section 17.11, “Enabling Secure or HTTPOnly Flags for Cluster Cookies,” on page 468](#)
- ♦ [Section 17.12, “Apache Portable Runtime Native Library Does Not Get Loaded in Tomcat,” on page 469](#)
- ♦ [Section 17.13, “Metadata Mentions Triple Des As Encryption Method,” on page 471](#)
- ♦ [Section 17.14, “Issue in Accessing Protected Resources with External Identity Provider When Both Providers Use Same Cookie Domain,” on page 471](#)
- ♦ [Section 17.15, “SAML Intersite Transfer URL Setup Does Not Work for Non-brokered Setups After Enabling SP Brokering,” on page 471](#)
- ♦ [Section 17.16, “Orphaned Identity Objects,” on page 472](#)
- ♦ [Section 17.17, “Users cannot Log In to Identity Provider When They Access Protected Resources With Any Contract Assigned,” on page 473](#)
- ♦ [Section 17.18, “Attribute Query from OIOSAML.SP Java Service Provider Fails with Null Pointer,” on page 473](#)
- ♦ [Section 17.19, “Disabling the Certificate Revocation List Checking,” on page 473](#)
- ♦ [Section 17.20, “Step up Authentication for the Identity Server Initiated SSO to External Provider Does not Work Unless It has a Matching Local Contract,” on page 473](#)
- ♦ [Section 17.21, “Metadata Cannot be Retrieved from the URL,” on page 473](#)
- ♦ [Section 17.22, “Requesting the Authentication to a Service Provider Fails,” on page 474](#)

- ♦ [Section 17.23, “SAML 2.0 POST Compression Failure Does not Throw a Specific Error Code,” on page 474](#)
- ♦ [Section 17.24, “SAML 1.1 Service Provider Re-requests for Authentication,” on page 474](#)
- ♦ [Section 17.25, “Issue in Generating WS-Federation Claim for SharePoint 2010 On Windows,” on page 474](#)
- ♦ [Section 17.26, “LDAP Authentication Fails while Accessing the Secret Store,” on page 475](#)
- ♦ [Section 17.27, “The Identity Server Statistics Logs Do Not Get Written In Less Than One Minute,” on page 475](#)
- ♦ [Section 17.28, “No Error Message Is Written in the Log File When an Expired Certificate Is Used for the X509 Authentication,” on page 476](#)
- ♦ [Section 17.29, “Terminating an Existing Authenticated User from the Identity Server,” on page 476](#)
- ♦ [Section 17.30, “Clustered Nodes Looping Due to JGroup Issues,” on page 477](#)
- ♦ [Section 17.31, “Authentication With Aliases Fail,” on page 477](#)

For information about Identity Server logging, see [Section 16.3, “Configuring Component Logging,” on page 414](#) and [Section 16.4, “Configuring Session-Based Logging,” on page 416](#).

## 17.1 Useful Networking Tools for 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 [Chapter 1, “Configuring an Identity Server,” on page 17](#).
- ♦ **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.

## 17.2 Troubleshooting 100101043 and 100101044 Liberty Metadata Load Errors

The Identity Server is the identity provider for 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 ensure 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 17.2.1, "The Metadata," on page 457](#)
- ♦ [Section 17.2.2, "DNS Name Resolution," on page 458](#)
- ♦ [Section 17.2.3, "Certificate Names," on page 459](#)
- ♦ [Section 17.2.4, "Certificates in the Required Trust Stores," on page 460](#)

If these steps do not solve your problem, try the following:

- ♦ [Section 17.2.5, "Enabling Debug Logging," on page 461](#)
- ♦ [Section 17.2.6, "Testing Whether the Provider Can Access the Metadata," on page 463](#)
- ♦ [Section 17.2.7, "Manually Creating Any Auto-Generated Certificates," on page 463](#)
- ♦ 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>).

## 17.2.1 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 457](#)
- ♦ ["Service Provider Metadata" on page 458](#)

### 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, 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 7.9, "Managing Metadata," on page 227](#).

### 17.2.2 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/spssoap
</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 462](#).

## 17.2.3 Certificate Names

Ensure that the certificates for the Identity Server and the Embedded Service Provider match the hostnames defined in the metadata URL (see [Section 17.2.2, “DNS Name Resolution,” on page 458](#)).

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 about how to create a certificate for the Identity Server, see [“Enabling SSL Communication” in the \*NetIQ Access Manager 4.0 SP1 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 about how to create an Access Gateways certificate, see [“Configuring the Access Gateway for SSL and Other Security Features” in the \*NetIQ Access Manager 4.0 SP1 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 463](#).

## 17.2.4 Certificates in the Required Trust Stores

Ensure 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. By default, well known external trusted certificates are bundled with Access Manager. You can view this list here: **Administration Console > Security > Certificates > External Trusted Roots**. If the issuer of server certificate is not present in the External Trusted Root list, the import the issuers of the server certificate (intermediate and trusted roots) 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.

For more information, see “[Importing a Signed Certificate](#)” in the *NetIQ Access Manager 4.0 SP1 Administration Console Guide*.

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 17.2.3, “Certificate Names,”](#) on page 459.
- 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 ensure that 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 ensure that 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 Servers 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 462](#).

## 17.2.5 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/nam/logs/idp` 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 462](#)
- ♦ [“Trusted Roots Are Not Imported into the Appropriate Trusted Root Containers” on page 462](#)
- ♦ [“The Server Certificate Has an Invalid Subject Name” on page 463](#)

## 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>
```

### 17.2.6 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
```

### 17.2.7 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.



## 17.3 Authentication Issues

This section discusses the following issues that occur during authentication:

- ♦ [Section 17.3.1, “Authentication Classes and Duplicate Common Names,” on page 464](#)
- ♦ [Section 17.3.2, “General Authentication Troubleshooting Tips,” on page 464](#)
- ♦ [Section 17.3.3, “Slow Authentication,” on page 465](#)
- ♦ [Section 17.3.4, “Federation Errors,” on page 465](#)
- ♦ [Section 17.3.5, “Mutual Authentication Troubleshooting Tips,” on page 465](#)
- ♦ [Section 17.3.6, “Browser Hangs in an Authentication Redirect,” on page 466](#)
- ♦ [Section 17.3.7, “Duplicate Set-Cookie Headers,” on page 466](#)

### 17.3.1 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 about creating custom login pages that prompt for more than username and password, see [Section 2.1, “Customizing the Identity Server Login Page,” on page 61](#).

### 17.3.2 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, ensure that 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 111](#).
- ♦ Check the user store health and replica layout. See [TID 3066352 \(http://www.novell.com/support/viewContent.do?externalId=3066352&sliceId=1\)](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.



- ♦ Ensure that 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.

### 17.3.3 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 *NetIQ Access Manager 4.0 SP1 Installation Guide*.
- ♦ If your LDAP user store is large, ensure that the search contexts are as specific as possible to avoid searching the entire tree for a user.

### 17.3.4 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.

### 17.3.5 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.

### 17.3.6 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.

### 17.3.7 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.

## 17.4 Problems Reading Keystores after Identity Server Re-installation

This can occur if you replace a hard drive and incorrectly reinstall the Identity Server. See [“Uninstalling Components”](#) in the *NetIQ Access Manager 4.0 SP1 Installation Guide* for the correct procedure.

## 17.5 After Setting Up the User Store to Use SecretStore, Users Report 500 Errors

If your eDirectory user store is running on SLES 11 SP1 64-bit (or a higher version) 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 \(http://www.novell.com/support/viewContent.do?externalId=7006437&sliceId=1\)](http://www.novell.com/support/viewContent.do?externalId=7006437&sliceId=1).

## 17.6 When Multiple Browser Logout Option Is Enabled User Is Not Getting Logged Out From Different Sessions

Allow multiple browser session logout option in the Identity Server cluster specifies 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. If you 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 and for logout URL you have to configure it as `/nidp/app/logout`.

## 17.7 302 Redirect to 'RelayState' URL after consuming a SAML Response is being sent to an incorrect URL

After consuming a SAML response, the browser is redirected to an incorrect URL, check whether the relay state is URL encoded. To fix this issue, have the following entries in the `web.xml` file as indicated below:

```
(/opt/novell/nids/lib/webapp/WEB-INF/web.xml)

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

## 17.8 Configuring SAML 1.1 Identity Provider Without Specifying Port in the Login URL Field

While adding the identity provider, do not specify the login URL and clear the show card option. Use the login URL to access the service provider:

```
https://idp.sitea.novell.com/nidp/saml/idpsend?
```

```
PID = https://idp.siteb.novell.com/nidp/saml/metadata&
```

```
TARGET = https://idp.siteb.novell.com/nidp/app
```

In the identity provider, while adding the service provider, configure ID in the intersite transfer page. Configure the login URL with port number -2443 instead of the provider ID URL:

```
https://idp.sitea.novell.com:2443/nidp/saml/idpsend?
```

```
id = <idname>&TARGET=https://idp.siteb.novell.com:2443/nidp/app
```

## 17.9 Attributes are Not Available Through Form Fill When OIOSAML Is Enabled

To work around this issue, create a new attribute set with the OIOSAML mandatory attributes having remote attribute mapping as its OID equivalent and associate the attribute set to the identity provider configured at the SP.

## 17.10 Issue in Importing Metadata While Configuring Identity Provider or Service Provider Using Metadata URL

To work around this issue, the administrator has to manually copy the metadata by selecting the metadata text option while configuring identity provider or service provider. The metadata text can be obtained from the browser.

## 17.11 Enabling Secure or HTTPOnly Flags for Cluster Cookies

By default the Identity Server and ESP cluster cookies do not have any secure or HTTPOnly flags.

To set the cluster cookies in the Identity Server you must add the following parameter at the `NIDP web.xml` and restart Tomcat:

Add the following parameters in `web.xml` below the `IdapLoadThreshold` context parameter:

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

```
<context-param>

    <param-name>httponlyClusterCookie</param-name>
    <param-value>true</param-value>

</context-param>
```

To set the cluster cookies in ESP, you must add the following parameter at the NESP `web.xml` and restart Tomcat:

Add the following parameters in the `web.xml` below the `ldapLoadThreshold` context parameter:

```
<context-param>

    <param-name>httponlyClusterCookie</param-name>
    <param-value>true</param-value>

</context-param>
```

---

**NOTE:** The secure cookies cannot be configured for ESP cluster as the communication between the Access Gateway and NESP is over HTTP on the loopback interface.

---

## 17.12 Apache Portable Runtime Native Library Does Not Get Loaded in Tomcat

The Apache Portable Runtime (APR) native library is not enabled by default. To workaround this issue, enable the APR native library.

Steps to enable the APR native library:

1. Download OpenSSL from (<http://www.openssl.org/source/>).
2. Extract the source (`tar -zxvf openssl-version`).
3. Compile and install (`cd openssl-version`) using the `./config`, `./config shared`, `make`, and `sudo make install` commands.

For example:

```
idp:~/openssl-0.9.8q #./config
idp:~/openssl-0.9.8q #./config sharedapr-1.4.5
idp:~/openssl-0.9.8q #make
idp:~/openssl-0.9.8q #sudo make install
```

4. Download APR from (<http://apr.apache.org/download.cgi>).
5. Extract the source (`tar -zxvf apr-version`).
6. Compile and install (`cd apr-version`) using the `./configure`, `make`, and `sudo make install` commands.

For example:

```
idp:~/apr-apr-1.4.5 #./configure
idp:~/apr-apr-1.4.5 #make
idp:~/apr-apr-1.4.5 #sudo make install
```

7. Create a `lib` folder under `Openssl-version`. For example, `idp:~/openssl-0.9.8q/lib #`
8. Copy `*.so` files from `openssl-version` to `lib` using `idp:~/openssl-0.9.8q/lib #cp ../*.so`.

9. Extract the wrapper library sources located in the Tomcat binary bundle \$CATALINA\_HOME/bin/tomcat-native.tar.gz or download the latest version.
10. Extract the source, compile, and install \$CATALINA\_HOME/bin/tomcat-native-1.1.20-src using this command:

```
$CATALINA_HOME/bin/tomcat-native-1.1.20-src/jni/native# ./configure --with-apr=/apr-version
folder location from root --with-java-home=/jdk location from -- libdir=/usr/lib/lib64 --prefix=/usr/
lib/lib64 --with-ssl=/openssl folder version from root.
```

```
Example: Idp1:/var/opt/novell/tomcat7/bin/tomcat-native-1.1.20-src/jni/
native#./configure --with-apr=/root/apr-1.4.5 --with-java-home=/opt/novell/
jdk1.6.0_26/
```

If the message says "checking OpenSSL library version... ok", installation is successful. If it shows "checking OpenSSL library version... is not compatible", installation is not successful.

#### 11. Tomcat-Native-library compilation and installation:

```
idp1:/$CATALINA_HOME/bin/tomcat-native-1.1.20-src/jni/native # make
idp1:/$CATALINA_HOME/bin/tomcat-native-1.1.20-src/jni/native # sudo make
install
```

12. Go to idp:/usr/lib/lib64 #. Create link of these two files using the following command:

```
Idp:/usr/lib/lib64 # sudo ln -s libtcnative-1.dylib libtcnative-1.jnilib
```

13. Copy all files from Idp:\$CATALINA\_HOME/bin/tomcat-native-1.1.20-src/jni/native/.libs to your #Native library path (JAVA\_LIB\_PATH).

```
Idp:/var/opt/novell/tomcat7/bin/tomcat-native-1.1.20-src/jni/native/.libs # cp
* /opt/novell/lib64
```

14. If you are running the application through the APR connectors, it needs the SSL certificate file in the .pem format.

#### To Create SSL Certificate File :

- a. Log in to Admin Console.
- b. Under the **Security** tab, click Certificates.
- c. Select the certificate, which is configured in your IDP Cluster and click that certificate name.  
Example: Configure a certificate with the name [new\_idp] and click new\_idp.
- d. Click **Export Private/Public KeyPair**.
- e. Enter the encryption and decryption password.
- f. Save the file. File is saved as .pfx.
- g. Convert the .pfx format to .pem using <https://www.sslshopper.com/ssl-converter.html>.

**Certificate file to Convert:** Enter the location where you have saved the .pfx file.

**Type of Current Certificate:** Select PFX/PKCS#12.

**Type To Convert To:** select Standard PEM.

Click **ConvertCertificate**.

**PFX Password:** Enter the same password as given in step e and save the file.

15. If you need to run your Admin Console using the APR connector, configure the following details into server.xml of your Admin Console. (vi /opt/novell/nam/adminconsole/conf/server.xml)

If you need to run your Identity Provider Server using the APR connector you can configure these details into server.xml of your Identity Provider Server. (vi /opt/novell/nam/idp/conf/server.xml)

16. APR Lifecycle Listener <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />

17. APR Connector configuration:

Example: <Connector port="8443" maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75" enableLookups="false" UploadTimeout="true" acceptCount="100" scheme="https" secure="true" clientAuth="false" sslProtocol="TLS" SSLEngine="on" SSLCertificateFile="\${catalina.home}/tomcatcert.pem"

SSLCertificateKeyFile="\${catalina.home}/tomcatkey.pem" SSLPassword="tomcat" />

Give the SSL Certificate File the same name as you entered for the .pem file.

---

**NOTE:** SSLCertificateFile and sslProtocol are required to run when you are using the APR connectors.

---

18. Restart tomcat.

## 17.13 Metadata Mentions Triple Des As Encryption Method

The OIO SAML metadata has tripledес-cbc and AES128-cbc mentioned as encryption methods. If triple des is not required, edit the following related tag in metadata and import the metadata manually.

Node: <md:EncryptionMethod

Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>

## 17.14 Issue in Accessing Protected Resources with External Identity Provider When Both Providers Use Same Cookie Domain

To workaround this issue, set 'agm.lagmode=false' in /opt/novell/singlebox/mag/webapps/agm/agm.properties.

## 17.15 SAML Intersite Transfer URL Setup Does Not Work for Non-brokered Setups After Enabling SP Brokering

To workaround this issue:

- ♦ Create a brokering group that has local IDP as Identity Provider and SP1 and SP2 as Trusted Providers.
- ♦ Create brokering rules for the Intersite Transfer URL requests to SP2. All requests to SP1 will be allowed.

## 17.16 Orphaned Identity Objects

When a persistent federation is configured or a transient federation with user mapping is configured by using Liberty, SAML 1 and SAML 2.0, the federation objects are created in the configuration store.

When you delete or disable a user object, the objects in the configuration datastore related to this specific user become orphaned. These orphaned user profile objects affect the user lookup operations and system performances. These objects have to be removed manually using the Defed Tool: Federation Entry Management.

This tool clears all the orphaned federation objects related to Liberty, SAML 1, and SAML 2 from the trust and configuration datastore, except for Shared Secret entries.

---

**NOTE:** When the Access Manger setup includes Access Gateway and no persistent or transient federations have been configured, these objects are not created.

---

*Linux:*

- 1 Change the current working directory to `/opt/novell/devman/nam_tools/` from a terminal.
- 2 Run this command:  

```
/opt/novell/java/bin/java -classpath ../lib/nam_tool.jar:../lib/nidp.jar:../lib/NAMCommon.jar:../lib/bcprov-jdk15-140.jar -Djava.util.logging.config.file=../conf/logging.properties com.novell.nam.tools.defed.DefedTool
```
- 3 The Defed tool will ask either to delete the orphan objects or exit from the tool. Select the option to delete the orphan objects. The tool will ask you to provide the IP address, port, user DN, and password.
- 4 The Defed tool deletes the orphaned federation objects and gives the summary of total number of federation entries encountered and number of the federation objects deleted.  
This tool can be used to perform the operation on remote server too.

*Windows:*

- 1 Go to the `C:\Program Files (x86)\Novell\nam_tools` folder.
- 2 Run this command:  

```
C:\Program Files (x86)\Novell\nam_tools>java -cp lib/nam_tool.jar;lib/nidp.jar;lib/NAMCommon.jar;lib/bcprov-jdk15-140.jar -Djava.util.logging.config.file=conf\logging.properties com.novell.nam.tools.defed.DefedTool
```
- 3 Provide IP address, port, user DN, and password.
- 4 The Defed tool deletes the orphaned federation objects and gives the summary of total number of federation entries encountered and number of the federation objects deleted.  
This tool can be used to perform the operation on remote server too.



## 17.17 Users cannot Log In to Identity Provider When They Access Protected Resources With Any Contract Assigned

To workaround this issue, ensure that the **Show Card** option is enabled on the default contract.

## 17.18 Attribute Query from OIOSAML.SP Java Service Provider Fails with Null Pointer

To workaround this issue:

- 1 Enable the OIOSAML compliance with service provider. The OIOSAML attribute set will be populated.
- 2 By default, the mandatory attributes are listed in the **Available** list.
- 3 Ensure that these mandatory attributes are moved from the **Available** list to the **Send with authentication** list to avoid the Null Pointer exception with OIOSAML compliance service providers.

## 17.19 Disabling the Certificate Revocation List Checking

For ADFS 2.0 to work with NetIQ Access Manager SAML 2.0, it is required to disable the Certificate Revocation List (CRL) checking.

To disable the CRL checking:

- 1 Modify the `tomcat7.conf` file of the Identity Server located at `/opt/novell/nam/idp/conf/tomcat7.conf`
- 2 Add this parameter `JAVA_OPTS="${JAVA_OPTS} -Dcom.novell.nidp.serverOCSPCRL=false"`
- 3 Restart the Identity Server by using this command: `/etc/init.d/novell-idp restart`.

## 17.20 Step up Authentication for the Identity Server Initiated SSO to External Provider Does not Work Unless It has a Matching Local Contract

For example, if a service provider is configured for a satisfiable contract that is only satisfiable by an external provider, then Intersite transfer service does not work.

To workaround this issue, ensure that any local contract is associated with the service provider. If not, then associate the same. External provider without any local contract is not supported

## 17.21 Metadata Cannot be Retrieved from the URL

To workaround this issue, verify the network card configuration for the proper DNS.

## 17.22 Requesting the Authentication to a Service Provider Fails

To workaround this issue:

- 1 In the Administration Console, click **Devices > Identity Servers > Edit > SAML 2.0 > [Service Provider] > Authentication Response**.
- 2 Change **Artifact** to **Post** in the **Binding** field.

## 17.23 SAML 2.0 POST Compression Failure Does not Throw a Specific Error Code

The POST Compression feature is supported when both the identity provider and service provider understand SAML 2 POST deflate and inflate. If the service provider sends a compressed message, the identity provider needs to decompress the message and vice-versa. For the Access Manager identity server and service provider, the `nidpconfig.properties` file located in `/opt/novell/nam/idp/webapps/nidp/WEBINF/classes` needs to be modified to enable the SAML 2.0 POST deflate and inflate.

## 17.24 SAML 1.1 Service Provider Re-requests for Authentication

The behavior of SAML 1.1 service provider has been changed in 3.1 Access Manager to perform a strict check on the name space of the attributes received in assertion.

To disable this, configure the following parameter in `web.xml`:

```
<context-param>
  <param-name>saml1xAttributeMatchByName</param-name>
  <param-value>true</param-value>
</context-param>
```

## 17.25 Issue in Generating WS-Federation Claim for SharePoint 2010 On Windows

When Access Manager is installed on Windows 2008 and uses a SharePoint 2010 WS-federation service provider to build trust relationship with the Identity Server, users get a 500 internal error while trying to log in to SharePoint 2010.

This error occurs because Access Manager does not find the method to generate the SharePoint claim after authenticating the user at the Identity Server.

To resolve this issue, perform the following steps:

- 1 Navigate to `C:\Program Files (x86)\Novell\Tomcat\bin`.
- 2 Open the `tomcat7w` file.
- 3 Navigate to the **Java** tab.
- 4 Search for the following Java Options placeholder:

```
-Djava.endorsed.dirs=C:\PROGRA~2\Novell\Tomcat\common\endorsed
```

5 Modify this option as follows:

```
-Djava.endorsed.dirs=C:\PROGRA~2\Novell\Tomcat\endorsed
```

---

**NOTE:** Check for the endorsed folder in Tomcat (Catalina Home). This folder should contain `stax-api-1.0.1.jar`, `serializer.jar`, `xalan.jar`, `xercesImpl.jar`, and `xml-apis.jar`.

---

## 17.26 LDAP Authentication Fails while Accessing the Secret Store

When a secret store is configured to use an external LDAP user store, the Access Gateway returns an error. The Identity Provider DN value set by the Administration Console for user stores enabled for NMAS is different from what the Identity Provider expects.

This issue is not applicable for new NMAS configurations as the Administration Console writes the correct value. If you have configured NMAS before upgrading to 3.2 SP2 IR1, follow either of the two workarounds:

### Workaround 1:

1. Apply the 3.2 SP2 IR1 patch.
2. Go to **Identity Server > Edit Cluster > Local > User Store**.
3. Clear **Install NMAS SAML Method**, click **Apply**, and **Update** the server.
4. Select **Install NMAS SAML Method**, click **Apply** and **Update** the server again.

### Workaround 2:

1. Go to **Roles and Tasks > NMAS > NMAS Login Methods > SAML Assertion > Affiliates**.
2. Click the cluster ID and set **Provider ID** as `<Cluster Base URL>/nidp/idff/metadata`. For example, `http://suse-87-129.blr.novell.com:8080/nidp/idff/metadata`.

## 17.27 The Identity Server Statistics Logs Do Not Get Written In Less Than One Minute

The Identity Server statistics logs do not get written before one minute even though the time specified is less than one minute, for example, 10 seconds. This issue happens only when the time is specified to less than one minute.

Do not specify the time less than one minute. As a best practice, you should not set small period because it increases the load on the server and also increases the log file size exponentially.

## 17.28 No Error Message Is Written in the Log File When an Expired Certificate Is Used for the X509 Authentication

When a user tries to authenticate with an expired client certificate, the authentication fails. The log file does not have any information about the expiration of the certificate. Browsers also do not display any error message about it.

To see the logs related to expired certificates, perform the following steps:

- 1 Enable the following Java option in `tomcat7.conf` under `/opt/novell/nam/idp/conf/`:

```
JAVA_OPTS="$ {JAVA_OPTS} -Djavax.net.debug=ssl,handshake"
```

This option enables SSL logs.

- 2 Restart the Identity Server.

## 17.29 Terminating an Existing Authenticated User from the Identity Server

Access Manager provides the ability for users to single sign on to back end web servers. These back end web servers provide a series of protected resources that users can only access once authenticated to an Identity Server, and authorised by the Access Gateway. Having parsed the user credentials, and credentials validated against a back end user store, the Identity Server creates and maintains an active session for that user. Only when the user manually logs out of the Identity Server, or if the user's session timeout expires, then the user's active session will be removed. If the user continuously accesses protected resources before the session timeout expires, the session can remain active forever.

**Use case:** You may want to terminate an authenticated user sometimes. Some of the cases are listed below.

- ♦ User A who currently has an active session on the Identity Server and access to many protected resources, has had a designation change within the organization causing a change to resources that may be available. By forcing user A to logout and login again, his new roles or attributes may be retrieved by the Identity Server and used in policy evaluations by Access Manager to reflect his new position.
- ♦ User B who currently has an active session on the Identity Server and access to many protected resources, has been asked to leave an organization and all access to protected resources must be removed. By terminating user B's session on the Identity Server, any subsequent requests to the Identity Server will require the user to login again.

The **User Sessions** page in the Administration Console helps you to find users logged into your system and also helps to terminate their sessions if required. It displays the active user details for each Identity Server. You can search for a user with the user ID and terminate the sessions.

- 1 In the Administration Console, click **Auditing > Troubleshooting > User Sessions**.
- 2 Specify the user ID and click **Search**. If a match is found, it lists the IP address of the Identity Server and its sessions.
- 3 Click **Terminate Sessions** to terminate the sessions of the specific user.

After you have performed the above procedure, the user sessions are terminated from the Identity Server, and any other trusted service providers it has provided an identity to during this session, for example, an Access Gateway or SAML 2.0 service provider.

---

**NOTE:** User details are fetched once per administration session. The last updated date is displayed. To refresh the data, click on **Refresh**.

---

## 17.30 Clustered Nodes Looping Due to JGroup Issues

In a cluster when multiple nodes are down, failover does not occur and user experiences looping due to jgroups issues.

**Workaround:** Modify the `/opt/novell/nids/lib/webapp/WEB-INF/web.xml` file on the Identity Server as follows to increase the jgroup timeouts.

```
<context-param>

  <param-name>JGroupsConfiguration</param-name>

  <param
value>TCP(start_port=[nidp:ClusterPort];end_port=[nidp:ClusterPort][nidp:IfExternalAddress];external_addr=[nidp:ExternalAddress][nidp:EndIf]):TCPPING(initial_hosts=[nidp:ClusterMembers];port_range=1;timeout=3500;num_initial_members=2;up_thread=true;down_thread=true):MERGE2(min_interval=10000;max_interval=30000):FD_SOCK([nidp:IfExternalAddress]bind_addr=[nidp:ExternalAddress][nidp:EndIf]):FD(shun=true;timeout=5000;max_tries=5;up_thread=true;down_thread=true):VERIFY_SUSPECT(timeout=2000;down_thread=false;up_thread=false):pbcast.NAKACK(down_thread=true;up_thread=true;gc_lag=100;retransmit_timeout=3000):pbcast.STABLE(desired_avg_gossip=20000;down_thread=false;up_thread=false):pbcast.STATE_TRANSFER():pbcast.GMS(merge_timeout=10000;join_timeout=5000;join_retry_timeout=2000;shun=true;print_local_addr=[nidp:DebugOn];down_thread=true;up_thread=true)</param-value>

</context-param>
```

For more information about the timeout options, see the following links:

- [TCPPING](#)
- [MERGE2](#)
- [FD\\_SOCK](#)
- [FD](#)
- [VERIFY\\_SUSPECT](#)
- [NAKACK](#)
- [pbcast.STATE\\_TRANSFER](#)
- [pbcast.GMS](#)

## 17.31 Authentication With Aliases Fail

If your userstore contains alias users and if you have configured alias class for authentication, authentication fails.

**Workaround:** For the workaround, see [TID 7015163](#).



---

# A About Liberty

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 \(http://www.projectliberty.org/\)](http://www.projectliberty.org/).

Liberty resources, including specifications, white papers, FAQs, and presentations, can be found at the [Liberty Alliance Resources Web site \(http://www.projectliberty.org/liberty/resource\\_center/\)](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	<a href="http://www.projectliberty.org/">Liberty Alliance Project Overview (http://www.projectliberty.org/)</a>
Liberty White Papers	<a href="http://www.projectliberty.org/liberty/resource_center/papers">Papers (http://www.projectliberty.org/liberty/resource_center/papers)</a>
Identity Federation Specifications	<a href="http://www.projectliberty.org/resources/specifications.php#box1">Liberty ID-FF 1.2 Specification (http://www.projectliberty.org/resources/specifications.php#box1)</a>
Web Service Framework Specifications	<a href="http://www.projectliberty.org/resources/specifications.php#box2a">Liberty ID-WSF 1.1 Specifications (http://www.projectliberty.org/resources/specifications.php#box2a)</a>
Liberty Profile Service Specifications	<a href="http://www.projectliberty.org/resources/specifications.php#box3">Liberty Alliance ID-SIS 1.0 Specifications (http://www.projectliberty.org/resources/specifications.php#box3)</a>
OASIS Standards (SAML)	<a href="http://www.oasis-open.org/specs/index.php#samlv2.0">Oasis Standards (http://www.oasis-open.org/specs/index.php#samlv2.0)</a>





---

# B Understanding How Access Manager Uses SAML

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) (<http://www.oasis-open.org/specs/index.php>).

This section describes how Access Manager uses SAML. It includes the following topics:

- [Section B.1, “Attribute Mapping with Liberty,” on page 481](#)
- [Section B.2, “Trusted Provider Reference Metadata,” on page 482](#)
- [Section B.3, “Identity Federation,” on page 482](#)
- [Section B.4, “Services,” on page 482](#)
- [Section B.5, “What's New in SAML 2.0?,” on page 482](#)
- [Section B.6, “Identity Provider Process Flow,” on page 483](#)
- [Section B.7, “SAML Service Provider Process Flow,” on page 485](#)

## B.1 Attribute Mapping with Liberty

Attribute-based 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 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 191.](#))

You can find detailed information about SAML 2.0 on the [OASIS Standards Web site](http://www.oasis-open.org/specs/) (<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 203.)

---

**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 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 NetIQ product terminology, a PEP could be thought of as the NetIQ Access Gateway, and the PDP as the NetIQ Identity Server.

## B.5 What's New in SAML 2.0?

SAML 2.0 provides several new features:

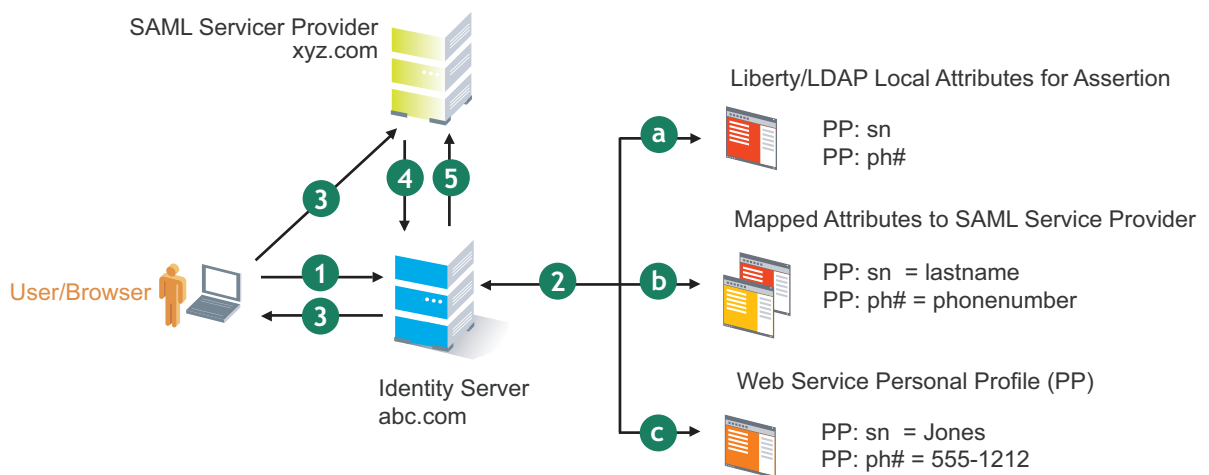
- ♦ **Pseudonyms:** An arbitrary name assigned by the identity provider to identify a user to a service provider. The identifier has meaning only in the context of the relationship between the relying parties. They can be a principal's e-mail or account name. Pseudonyms are a key privacy feature that 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 15.6, "Mapping LDAP and Liberty Attributes," on page 397](#).) 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 *phonenum**ber*, respectively. (See [Section 7.8.1, “Configuring the Attributes Obtained at Authentication,” on page 224.](#))

- 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.

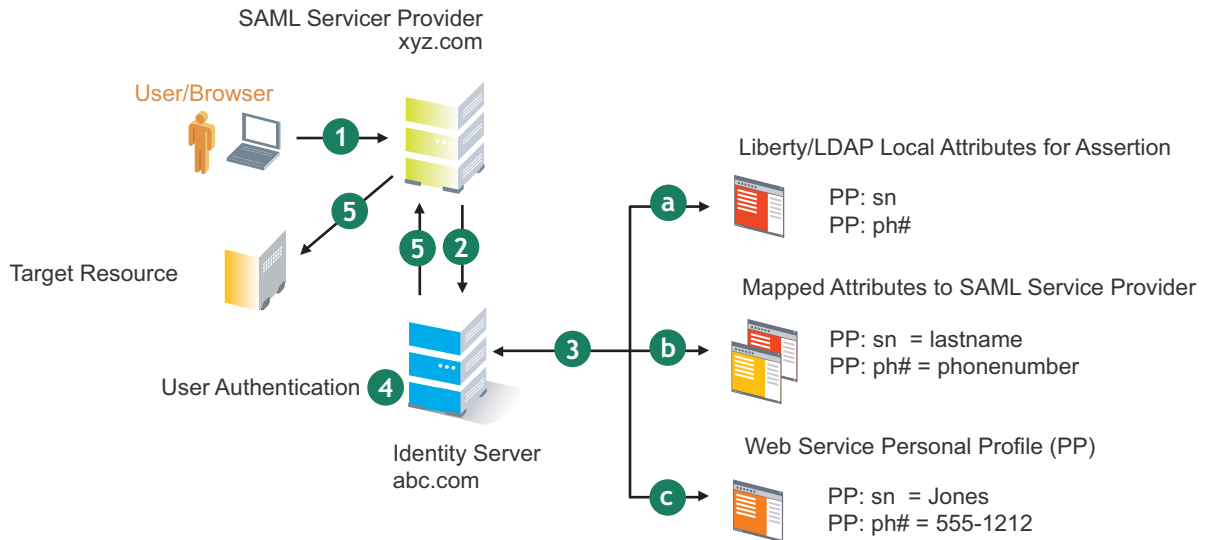
The remote SAML server receives the artifact and looks up the assertion. The assertion is sent to the SAML server at *xyz.com* in a SOAP envelope. The assertion contains the attributes *lastname=Jones* and *phonenum**ber=555-1212*.

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 *phonenum=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, you specified that *PP: sn* and *PP: ph#* should be used.

- a. The Identity Server processes the Liberty attribute map (see [Section 15.6, "Mapping LDAP and Liberty Attributes," on page 397](#)) to the SAML implementation-specific attributes (see [Section 7.8.1, "Configuring the Attributes Obtained at Authentication," on page 224](#)).

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 *phonenum*, 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.

---

# C Data Model Extension XML

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 15.2.1, “Modifying Service and Profile Details for Employee, Custom, and Personal Profiles,” on page 387](#)). 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>
</SchemaExtensions>
```

## C.1 Elements

The definition of the attributes for each data model extension XML element are as follows:

- ♦ [“Root Element” on page 488](#)
- ♦ [“Group Element” on page 488](#)
- ♦ [“Extension Element” on page 489](#)
- ♦ [“ValueSet Element” on page 490](#)
- ♦ [“Value Element” on page 490](#)

## 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/ep:CorpLegalIdentity/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:

- ♦ CaseIgnore
- ♦ 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:Group> </wsfc:Root>
  </wsfc:SchemaExtensions>
```

```

</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>

```



---

# D Configuring the Supported Social Authentication Providers for API Keys and API Secrets

Access Manager requires API Keys and API Secrets from the supported social authentication providers to integrate with these providers. Follow the steps to configure the supported applications and to get keys from the social authentication providers. You can integrate with Facebook, LinkedIn, Twitter, and Google+. For other providers, see [Section 4.1.3, “Configuring SocialAuthClass,” on page 146](#).

---

**NOTE:** The procedures documented below may not match the Social Networking Providers’ interface when you create an application. If there are any changes, follow the wizard accordingly. The procedure below is for reference purpose and can vary based on provider configuration page.

---

## D.1 Integrating Access Manager with Facebook

The following procedure enables you to generate API Key and API Secret with Facebook.

- 1 Create a Facebook application for community.
  - 1a Log in to Facebook to access the Application page: <https://developers.facebook.com/apps>.
  - 1b From the top right corner, click **Create New App**.
  - 1c Fill in the following fields in the Create a new app screen:
    - ♦ **Display Name:** Specify a name for web application.
    - ♦ **Category:** Select a category from the drop-down list.
  - 1d Click **Create App**.
  - 1e In the Security Check page key in the displayed Captcha text in the **Text in the box** field and click **Submit**.
  - 1f The Dashboard page displays App Name, App ID and App Secret (hidden). Click **Show** to display the **App Secret**.
  - 1g Copy the values of **App ID** and **App Secret** parameters. You will need these values when you configure Facebook with Access Manager.
  - 1h Click **Settings** on the left. In the **Basic** tab, specify **Contact Email** address.
  - 1i Click **Advanced** tab and specify the following details:
    - ♦ Deauthorize Callback URL - For example: `https://<IDP URL>:<Port Number>/nidp/app`
    - ♦ Valid OAuth redirect URIs - For example: `https://<IDP URL>:<Port Number>/nidp/jsp/socialauth_return.jsp`
  - 1j Click **Status & Review**.

- 1k Select **YES** to make this application and all its live features available. By default, it is selected as **No**.
      - 1l Your application status will change to Green and will be available online.
- 2 Configure Facebook application Configuration Setting in Access Manager. The **App ID** and **App Secret** will be used by Access Manager to configure Facebook.

## D.2 Integrating Access Manager with LinkedIn

The following procedure enables you to generate API Key and API Secret with LinkedIn.

- 1 Create a LinkedIn application for community.
  - 1a Log in to LinkedIn to access the Developer Network page: <https://www.linkedin.com/secure/developer>.
  - 1b Click **Add New Application**.
  - 1c Fill in the following fields in the form displayed:
    - ♦ **Company Info:** Select **New Company** from the drop-down list and specify the name of the company.
    - ♦ **Application Info:** Specify the **Application Name**, **Description**, **Website URL** where people will learn about application and select **Application Use** and **Live Status**.
    - ♦ **Contact Info:** Specify the developer contact email and phone number.
    - ♦ **OAuth User Agreement:** Specify the OAuth URLs and agreement language.
    - ♦ Accept **Terms of Service** and **Add Application**. A message that your application was successfully created appears.
  - 1d Copy the value of **API Key** and **Secret Key** parameters. These values will be required when you configure LinkedIn providers with Access Manager.
- 2 Configure LinkedIn application Configuration Setting in Access Manager. The **API Key** and **Secret Key** will be used by Access Manager to configure LinkedIn.

## D.3 Integrating Access Manager with Twitter

The following procedure enables you to generate API Key and API Secret with Twitter.

- 1 Create a Twitter application for community.
  - 1a Log in to Twitter to access the Developers page using Twitter credentials: <https://dev.twitter.com/>.
  - 1b From the drop-down box on the right, click on **My Applications**.
  - 1c From Twitter Apps page, click **Create New App** to display Create an application page.
  - 1d Fill in the following fields in the **Application details** page:
    - ♦ Specify the application **Name**, **Description**, **Website URL**, for example, <http://Yourdomain:port> and **Callback URL**, for example, [http://Yourdomain:port/nidp/jsp/socialauth\\_return.jsp](http://Yourdomain:port/nidp/jsp/socialauth_return.jsp).
  - 1e Select **Yes, I agree** and click **Create your Twitter application**.
  - 1f From the Details page, verify the details of this application.
  - 1g Navigate to **Settings** tab, select **Allow this application to be used to Sign in with Twitter** and click **Update settings**.

- 1h Navigate to **API Keys** tab, copy the values of **API key** and **API secret** parameter. These values will be required when you configure Twitter providers with the Access Manager.
  - 1i In your access token, click **Create my access token**. A message that your access token has been successfully created appears.
- 2 Configure Twitter application Configuration Setting in Access Manager. Access Manager uses the **API key** and **API secret** to configure Twitter.

## D.4 Integrating Access Manager with Google+

The following procedure enables you to generate API Key and API Secret with Twitter.

- 1 Create a Google+ application for community.
  - 1a Log in to Google+ API console. Sign in to <https://code.google.com/apis/console> (<https://code.google.com/apis/console>) using Google+ credentials.
  - 1b From **API Project** on the left select **Create** from the drop-down list.
  - 1c From Create project page, click **Create project**.
  - 1d On the All Services page, in the left pane, click **API Access**.
  - 1e To create an application, on the API Access page, click **Create an OAuth 2.0 client ID...**
  - 1f In the Create Client ID page, in the **Product name** field, specify a valid name and click **Next**.
  - 1g On Client ID Settings page, for **Application type**, choose **Web application**, if it is not selected by default.
  - 1h In **Your site or hostname**, specify the URL of your production server. For example, `example.namdemo.com:8443`.
  - 1i Click **Create client ID**. To generate **Client ID** and **Client secret** for your application, click **Edit settings**.
  - 1j In the Edit client settings page, specify the authorized **Redirect URIs**, enter `https://example.namdemo.com:8443/nidp/jsp/socialauth_return.jsp`.
  - 1k In the authorized **JavaScript origins**, enter: `https://example.namdemo.com:8443` and click **Update**.
- 2 Configure Google+ application Configuration Setting in Access Manager. Access Manager uses the **Client ID** and **Client secret** to configure Google+.

