Novell SAML Extension for Novell iChain®

1.0

www.novell.com

ADMINISTRATION GUIDE

January 26, 2005

Novell®

Legal Notices

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

www.novell.com

SAML Extension 1.0 for Novell iChain
January 26, 2005

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

ConsoleOne is a registered trademark of Novell, Inc. in the United States and other countries.
eDirectory is a trademark of Novell, Inc.
iChain is a registered trademark of Novell, Inc. in the United States and other countries.
NetWare is a registered trademark of Novell, Inc. in the United States and other countries.
Novell is a registered trademark of Novell, Inc. in the United States and other countries.
Novell Certificate Server is a trademark of Novell, Inc.

Third-Party Materials

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

Contents

About This Guide

The purpose of this documentation is to help you install the components of the SAML extension for Novell® iChain® software.

The audience for this documentation is network administrators.

This guide is divided into the following sections.

- Chapter 1, "SAML Overview," on page 9 — An explanation of the benefits of the SAML extension for Novell iChain and an overview of the components that make up the SAML extension product.
- Chapter 2, "Installing the SAML Extension Software," on page 27 — Instructions for how to install the SAML extension components, including system requirements and software installation instructions.
- Chapter 3, "Configuring the SAML Extension," on page 39 — Instructions on how to configure the SAML extension with ConsoleOne Snap-ins.
- Chapter 4, "Configuring the SAML Extension Server," on page 67 — Instructions on how to configure the SAML extension server.
- Chapter 5, "SAML Security Considerations," on page 71 — Information on the SAML extension server's security components and how to configure them.
- Chapter 6, "Enabling Web Sites With SAML Single Sign-On Functionality," on page 101 — Web developer information on how to use SAML's single sign-on features.
- Appendix A, "Installing the Java Development Kit," on page 105 — Instructions for how to install the JDK.
- Appendix B, "Installing Tomcat Light Edition," on page 107 — Instructions for how to install and validate Tomcat 4.0 LE.

Documentation Conventions

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

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

1 SAML Overview

This section discusses the Security Assertions Markup Language (SAML), including common SAML use cases, SAML assertions, and SAML protocol. The following topics are included:

- Defining SAML
- Defining SAML Assertions
- SAML Producer Consumer Model
- SAML Protocol
- SAML Bindings and Profiles
- Additional Transaction Profiles
- SAML Going Forward
- SAML Extension for Novell iChain

Defining SAML

SAML is an XML language intended to facilitate the exchange of authentication, attribute, and authorization information. Why invent a new language to do this? First, authentication and permissions information is shared in mostly proprietary ways. This means that in most cases, for two different systems to communicate security information, custom code must be developed. Second, Web-based applications need to be able to authenticate across domains more easily. Simply passing usernames and passwords is not sufficient for enterprise applications.

Common Web Application Scenarios

There are numerous ways SAML-based technology can add functionality to identity-based applications. A few examples are:

- A user is a member of a pharmaceutical research organization and is logged in to his or her research portal. The user should have access to a sister organization's application to obtain clinical trial data. The user should be able to access trial data for the only the drugs his or her organization is working with.
- A corporate admin user can access a supplier's Web site to order office supplies. The user is granted access to supplies for which he or she has been cleared to purchase and can only purchase a certain quantity of goods.
- A corporate portal user wants to enroll in the company's benefits program. The user selects a link to the benefits provider who can present the employee with the appropriate benefits options, and can automatically enroll the user into the desired programs.

SAML Use Cases

SAML was developed to handle three generate types of use cases:

- Single sign-on
- Authorization services
- Back-office transactions

Single Sign-On

This is the most intuitive SAML use case. A user logs in to a source site. The user selects a link at the source site, directing him or her to a secure resource on a destination site. The destination site is able to authenticate the user and provide the user with the secure resource.

Authorization Services

A user has authenticated to a site and/or application. The user access to a resource controlled by a Policy Enforcement Point (PEP). The PEP checks for user access to the desired resource. The user is either granted or denied access to the resource. SAML is used as the communication mechanism between the PEP and a Policy Decision Point (PDP). In Novell product terminology, a PEP could be thought of as iChain®, and the PDP as Novell® eDirectory™ or another service.

Back-Office Transactions

A corporate user has logged into his or her company portal and wants to order office supplies. The office supply company can authenticate the user and determine if the user has rights to order the supplies, as well as what the user's spending limit is.

SAML Requirements

SAML provides the following key features that make the above-mentioned use cases possible:

- A standard XML format

  An XML schema defining SAML assertions

- A standard message exchange protocol

  An XML schema defining SAML messages

- Bindings to transport protocols

  Rules describing how messages are transmitted over different communications protocols (HTTP, HTTPS)

SAML 1.0 Features

SAML is, in essence, an XML schema and encoding for expressing security information. SAML-encoded security information is referred to as an assertion. A SAML assertion can contain a number of different SAML statements containing information about authentication, attributes, and authorization.

SAML also includes an XML schema and encoding system for making SAML requests and responses. A system receiving a user with SAML information needs to request an assertion from the user's source site. A Policy Enforcement Point (PEP) that needs authorization information can make that request to a Policy Decision Point (PDP) using SAML.

Finally, SAML includes a set of bindings that define how SAML messages, requests and responses are transported over different messaging protocols.

SAML 1.0 was recently submitted and ratified by the OASIS organization. This specification contains SAML bindings to SOAP over HTTP and HTTPS, as well as Web SSO profiles.

As of when this document was produced, SAML 1.1 is still in development. One of the major additions to SAML 1.1 will be the inclusion of a SAML binding to WS-Security.

Defining SAML Assertions

A SAML assertion is a declaration of fact about a subject. SAML defines three types of assertions that can be made about a given subject:

- Authentication: The subject is authenticated using a given authentication assertion method at a given time.
- Attribute: The subject has a given attribute.
- Authorization: The subject should or should not be granted access to a given resource.

The SAML specification is designed so that extensions can be made to include new types of information in an assertion. Assertions might or might not (depending on the transport mechanism) be digitally signed by their issuer.

Defining SAML Assertion Attributes

All assertions contain a number of required attributes:

- **Issuer identifier:** An issuer name that allows the receiver of the assertion to know who it came from.
- **Issuance time stamp:** Indicates the exact time the assertion was generated.
- **Assertion ID:** A unique identifier for the assertion.
- **Subject:** A name along with a security domain. The subject can also contain confirmation data.
- **Conditions:** A set of conditions under which the assertion should be accepted. SAML includes a built-in condition called a validity period that indicates when the assertion becomes valid and when it ceases to be valid.
- **Optional Advice:** Discretionary information that can be ignored by the Relying Party.

In order to be useful, SAML assertions must “assert” something. Statements about a user's authenticity, attributes, or authorization are placed under the main assertion element. An assertion must contain at least one statement.

SAML Authentication Statements

An authentication statement asserts that a subject S authenticated to the issuing system using method M at time T. A Relying Party receiving an assertion containing an authentication statement can choose to accept the user authentication performed at the issuer’s site. The authentication statement provides SAML SSO functionality.

The example below shows a simple SAML authentication assertion. For this example, the assertion does not contain any of the optional SAML data, nor does it declare the required SAML

namespace. The assertion states that user cn=joe,o=novell authenticated to Novell at the specified time using the password method.

Figure 1 Simple SAML Authentication Assertion

```
<saml:Assertion MajorVersion="1"
            MinorVersion="0"
            AssertionID="some_unique_identifier"
            Issuer="novell_issuer_id"
            IssueInstant="DATE_TIME"
<saml:AuthenticationStatement
      AuthenticationMethod="urn:oasis:names:tc:SAML: 1.0: am:p
      AuthenticationInstant="DATE_TIME">
   <saml:Subject>
         <saml:NameIdentifier>cn=joe,o=novell</saml:NameIdent
   </saml:Subject>
  </saml:AthenticationStatement>
</saml:Assertion>
```

SAML Attribute Statements

An attribute statement asserts that a subject S has attribute A in namespace N with value(s) V. A Relying Party can associate the subject with the provided attribute. This allows a Relying Party to create customize its application for external users. It also allows users accessing a site using SAML to be auto-provisioned with appropriate user data.

The example below shows a simple SAML attribute assertion. For this example, the assertion does not contain any of the optional SAML data, nor does it declare the required SAML namespace. The assertions states that user cn=joe,o=novell has attribute Email in namespace urn:test:attributes with value joe@novell.com.

Figure 2 Simple SAML Attribute Assertion

```
<saml:Assertion MajorVersion="1"
            MinorVersion="0"
            AssertionID="some_unique_identifier"
            Issuer="the_issuer_identifier"
            IssueInstant="DATE_TIME"
  <saml:AttributeStatement
    <saml:Subject>
          <saml:NameIdentifier>cn=joe,o=novell</saml:NameIdent
    <saml:Subject>
    <saml:Attribute AttributeName="Email"
              AttributeNamespace="urn:test:attributes">
    </saml:AttributeValue>joe@novell.com</saml:AttributeValue>
    <saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

SAML Authorization Statements

Authorization decision assertions are issued by a Policy Decision Point and are used to determine if a subject S should be granted access to a resource R with access type A, given the provided evidence E. The subject can be any entity, either a human user or a program, that wants access to a resource. The resource can represent any data or service that requires access control.

The example below shows a simple SAML authorization decision assertion. For this example, the assertion does not contain any of the optional SAML data, nor does it declare the required SAML namespace. The assertion states that user cn=joe,o=novell should be granted access to the Buy action on the uri:novell.com/buy_stuff resource.

Figure 3 Simple SAML Authorization Decision Assertion

```
<saml:Assertion MajorVersion="1"
            MinorVersion="0î
            AssertionID="some_unique_identifier"
            Issuer="the_issuer_identifier"
            IssueInstant="DATE_TIME"
  <saml:AuthorizationDecisionStatement
     Resource="uri:novell.com/buy_stuff"
     Decision="saml:Grant">
    <saml:Subject>
          <saml:NameIdentifier>cn=joe,o=novell</saml:NameIdent
    </saml:Subject>
    <saml:Action>Buy</saml:Action>
  </saml:AuthorizationDecisionStatement>
</saml:Assertion>
```

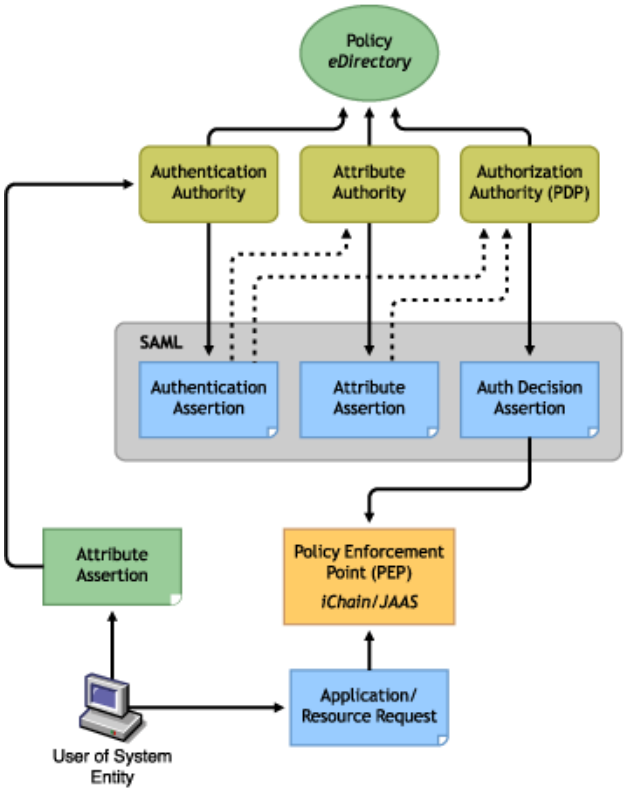
SAML Producer Consumer Model

Each different type of security information that SAML can express could be generated by a different authority. The conceptual model for SAML defines three separate authorities:

- **Authentication Authority:** Produces authentication information.
- **Attribute Authority:** Provides attribute information, and is analogous to the XACML PIP.
- **Authorization Authority or Policy Decision Point (PDP):** Provides authorization decision information and is analogous to the SACML PDP.

Figure 4 is a high-level diagram produced by the OASIS working group that illustrates the flow of information between these different authorities. Taken as a whole, the separate authorities can work together to provide a complete security infrastructure.

Figure 4 Information Flow Between Authorities

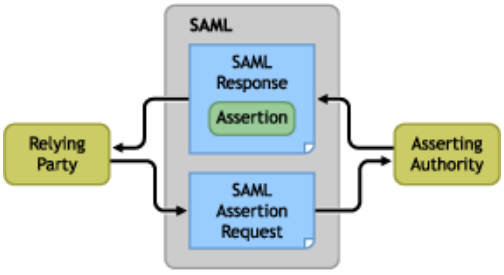


The model shows only a conceptual view of how a system could be put together. Each of the three authorities is not independent; each relies on other parts of the system to do its job. The authentication authority requires a credentials collector to provide it with authentication information. This could come in the form of a password login, a smart card, biometric, or a SAML authentication assertion produced by another authentication authority. The attribute authority relies on the authentication authority 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. The authorization authority or PDP relies upon authentication and attribute information in order to make its authorization decisions. The authorities are distinct and separate entities. Each part of the system has different roles, yet they could all be contained in a single service.

SAML Protocol

SAML defines a simple request response protocol that allows a Relying Party (a system entity that uses a security assertion) to request a SAML assertion from an asserting authority (a system entity that produces an assertion).

Figure 5 SAML Simple Request Response Protocol



In the SAML protocol, SAML assertions are encapsulated in a SAML Response. The intention is that systems with minimal knowledge of each other can interact using the SAML protocol.

SAML Requests

There are different types of requests for different types of SAML information. Each request type asks specific types of SAML information. The different types of requests defined by SAML 1.0 are:

- **Authentication Query:** Requests authentication information about a specified user.
- **Attribute Query:** Requests a set of attributes associated with a specific object.
- **Authorization Decision Query:** Requests a decision requiring access to a resource.
- **Artifact Query:** Requests a specific SAML assertion.This is usually an authentication assertion used for single sign-on (SSO).

Authentication Query

Authentication queries are used to determine if the authentication authority has authentication information about the provided subject. A successful response is in the form of assertions containing authentication statements. The authentication query is not used as a request for new authentication; it is only used to get information about previous interaction between the subject and the authentication authority.

In the example below, the SAMLP request makes a request for authentication information about the user cn=joe,o=novell. For this example, optional SAML information has been omitted along with the SAML and SAMLP namespace declarations.

Figure 6 Authentication Query

```
<samlp:Request MajorVersion="1"
               MinorVersion="0"
               RequestID="some_unique_identifier"
               IssueInstant="DATE_TIME"
  <samlp:AuthenticationQuery>
    <saml:Subject>
            <saml:NameIdentifier>cn=joe,o=novell</saml:NameIde
    </saml:Subject>
  </samlp:AuthenticationQuery>
</saml:Request>
```

Attribute Query

An attribute query is used to get attributes about a given subject. A successful response contains assertions that contain attribute statements about the subject.

In the example below, the SAMLP request makes a request for the Email attribute in the urn:test:novell namespace for the subject cn=joe,o=novell. For this example, optional SAML information has been omitted along with the SAML and SAMLP namespace declarations.

Figure 7 Attribute Query

```
<samlp:Request MajorVersion="1"
               MinorVersion="0"
               RequestID="some_unique_identifier"
               IssueInstant="DATE_TIME">
  <samlp:AttributeQuery>
    <saml:Subject>
            <saml:NameIdentifier>cn=joe,o=novell</saml:NameIde
    </saml:Subject>
    <saml:AttributeDesignator AttributeName="Email"
                              AttributeNamespace="urn:test:nov
    </saml:AttributeDesignator>
  </samlp:AttributeQuery>
</saml:Request
```

Authorization Decision Query

Authorization decision queries are used to determine if a subject S is allowed to perform action A on resource R.

In the example below, a sample authorization decision query asks if user cn=joe,o=novell should be able to perform the Buy action on the urn:novell/buy_stuff resource. For this example, the optional SAML information has been removed along with the SAML and SAMLP namespace declarations.

Figure 8 Authorization Decision Query

```
<samlp:Request MajorVersion="1"
               MinorVersion="0"
               RequestID="some_unique_identifier"
               IssueInstant="DATE_TIME">
  <samlp:AuthorizationDecisionQuery Resource="urn:novell/buy_s
    <saml:Subject>
            <saml:NameIdentifier>cn=joe,o=novell</saml:NameIde
    </saml:Subject>
    <saml:Action>Buy</saml:Action>
  </samlp:AuthorizationDecisionQuery>
</saml:Request
```

Artifact Query

An artifact request is a special type of SAMLP query used in the single sign-on (SSO) use case. The artifact query is used in the Browser/Artifact web SSO, which will be covered later. In the Browser/Artifact use case, a Relying Party is presented with an incoming request accompanied by a SAML Artifact. The SAML Artifact allows the Relying Party to determine who issued the user an Assertion and a unique identifier of that assertion. The Relying Party can then make an artifact query to the issuer that provides the assertion.

In the example below, a sample artifact query makes a request for an assertion with artifact SAMPLE_ARTIFACT. For this example, optional SAML data has been omitted along with the SAMLP and SAML namespace declarations.

Figure 9 Artifact Query

```
<samlp:Request MajorVersion="1"
               MinorVersion="0"
               RequestID="some_unique_identifier"
               IssueInstant="DATE_TIME">
  <samlp:AssertionArtifact>SAMLPLE_ARTIFACT</samlp:AssertionA
</saml:Request
```

SAML Responses

When a system receives a SAMLP request, a SAMLP response is returned. If the request can be successfully fulfilled then a SAMLP response is returned containing the requested assertions. If the SAMLP request cannot be fulfilled, a SAMLP response is returned containing an error message. The SAML 1.0 specification defines a number of static error codes that are associated with failed SAMLP requests.

The example below shows a sample of a successful SAMLP response:

Figure 10 Successful SAMLP Response

```
<samlp:Response MajorVersion="1"
                MinorVersion="0"
                ResponseID="some_unique_identifier"
                InResponseTo="ID_of_the_request"
                IssueInstant="DATE_TIME">
  <samlp:StatusCode>samlp:Sucess</samlp:StatusCode>
  ..<saml:Assertion>..
</saml:Request
```

A successful response contains at least one SAML assertion.

If there is an error fulfilling a SAMLP request, an error will be returned. The example below is a SAMLP response that describes an error condition. This example shows the error that is returned if the requester is using a request version that is too low.

Figure 11 Example of an Error in Fulfilling a SAMLP Request

```
<samlp:Response MajorVersion="1"
                MinorVersion="0"
                ResponseID="some_unique_identifier"
                InResponseTo="ID_of_the_request"
                IssueInstant="DATE_TIME">
  <samlp:StatusCode Value="samlp:Requester">
    </samlp:StatusCode Value="samlpRequestVersionTooLow">
  </samlp:StatusCode>
  <samlp:StatusMessage>Explanation of error</samlp:StatusMes
  <samlp:StatusDetail>Detail of error message</samlp:StatusDe
</saml:Response>
```

SAML 1.0 defines the following top-level status codes:

- Success
- VersionMismatch: Unable to fulfill the request because of an incorrect version.
- Requester: Unable to fulfill the request because of an error with the requester.
- Responder: Unable to fulfill the request because of an error with the responder.

SAML 1.0 also defines second-level status codes by providing additional information about why the SAMLP request failed:

- RequestVersionTooHigh
- RequestVersionTooLow
- RequestVersionDeprecated
- TooManyResponses: Fulfilling the request would require returning too much data.
- RequestDenied: The requester does not have sufficient rights to perform the request.
- ResourceNotRecognized: Unable to resolve the resource being referenced.

All of these error code values are QNames associated with the SAMLP namespace.

The StatusMessage and StatusDetail messages can contain any string data.

SAML Bindings and Profiles

The SAML 1.0 specification includes rules for binding SAMLP requests and responses to several communication protocols. Several Web SSO profiles are also defined by SAML 1.0. These are the Browser/Post and Browser/Artifact profiles.

SOAP Over HTTP Binding for SAML

Because SAMLP requests and responses are encoded in XML, their binding to SOAP is trivial: It amounts to the simple wrapping of SAMLP requests and responses into a SOAP envelope and body. The example below shows a SAMLP request and response transmitted over SOAP on HTTP. For this example, the namespace declarations have been omitted.

Figure 12 SAMLP Request and Response Transmitted Over SOAP on HTTP

```
POST/SamlService HTTP/1.0
Host: www.novell.com
Content-Type: text/xml
Content-Length: nnn

SOAPAction: http://www.oasis-open.org/.committees/security

<SOAP-ENV:Envelope><SOAP-ENV:Body >
  <samlp:Request> . . . </samlp:Request>
</SOAP-ENV:Envelop></SOAP-ENV:Body>
. .
. . .
. .

<SOAP-ENV:Envelop><SOAP-ENV:Body >
  <samlp:Request> . . . </samlp:Request>
</SOAP-ENV:Envelop></SOAP-ENV:Body>
```

Security

HTTPS provides message integrity and confidentiality. Optionally, authentication can be added by using client authentication. The SOAP over HTTP binding for SAML makes use of these properties to make communication between the issuing and relying parties. The SAML binding requires that the channel provide integrity and confidentiality, and makes authentication optional depending upon system requirements.

Web Browser SSO Profiles of SAML

The Web Browser SSO profile is arguably the most important use case of SAML. The Web Browser SSO Profile includes the following basic steps:

1. A user authenticates to a source site.
2. The user accesses an Intersite Transfer URL at the source site.
3. The destination site receives the user with SAML assertions.
4. The user obtains access to the desired resource.

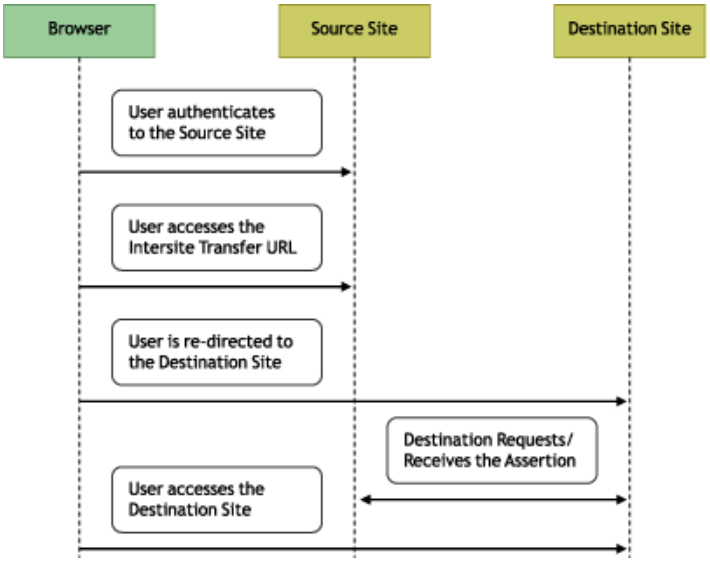
There are two SAML profiles defined to provide Web browser SSO:

- **Browser/Artifact:** Using a small amount of data on a redirect query string, a Relying Party can access the assertion by directly communicating with the issuing authority.
- **Browser/POST:** The entire assertion is transmitted to the Relying Party as a Form/POST.

Browser/Artifact Profile

Figure 13 is a basic activity diagram that illustrates the Browser/Artifact profile:

Figure 13 Basic Activity of Browser/Artifact Profile



The sequence of events is as follows:

1. The user authenticates to a source site using whatever authentication method the source site has in place.
2. The user clicks a link to the source site's Intersite Transfer URL. Clicking this link causes the source site to generate an authentication assertion for the user.
3. The user is redirected to the destination site's SAML Artifact Receiver URL. A SAML Artifact is included on the URL query string. The Target URL on the destination site the user wants to access is also included on the query string.
4. The destination site determines source of the artifact and calls back to the source site to retrieve the SAML authentication assertion.
5. The destination site validates the SAML authentication assertion and allows the user to access the desired resource.

There are several key things that happen in this profile that should be examined in more detail.

Intersite Transfer URL

The Intersite Transfer URL (Step 2 of the Browser/Artifact Profile section) is the resource at the source site that is responsible for generating a SAML assertion for the user. The format of the request to the Intersite Transfer URL is not defined by SAML 1.0. In practice, and in the Novell implementation, both the destination site ID and the destination target URL must be sent to on the query string. The destination site ID allows the Intersite Transfer service to know who it is generating the assertion for. Each different Relying Party site might have different rules for assertions that it accepts, so the Intersite Transfer service needs to take those into account when generating the assertion. Secondly, the Intersite Transfer service must know the actual Target URL the user wants to access on the destination site. After the Intersite Transfer service has generated and stored a SAML authentication assertion on behalf of the user, the user is redirected to the destination site. The user is not sent directly to the Target URL on the destination site, but rather, is sent to a special SAML Artifact Receiver URL at the destination site that is responsible for authenticating incoming SAML users. The redirect URL to the destination site's Artifact Receiver URL contains two pieces of information (as defined in SAML 1.0):

1. SAMLArtifact: [TypeCode (2 bytes)] [Source ID (20 bytes)] [Assertion Handle (20 bytes)] Base 64-encoded.
2. Target URL: A URL-encoded representation of the resource the user wants to access on the destination site.

Artifact Receiver URL

In Step 4 of the Browser/Artifact Profile section, the user is sent to the destination site's Artifact Receiver URL. The service running at this URL is responsible for authenticating incoming SAML users. The Artifact Receiver URL expects the SAMLArtifact and TargetURL query string parameters to be present on the incoming request or it fails. The Artifact Receiver URL first parses the SAMLArtifact to determine the ID of the assertion generator. Using that ID, the source site is able to determine what site issued the assertion associated with the incoming user. The source site is then able to send to that site a SAMLP Request for the assertion referenced by the received SAMLArtifact. The destination site can then validate the returned assertion. If the received assertion is valid, the destination site authenticates the user and redirects the user to the Target URL.

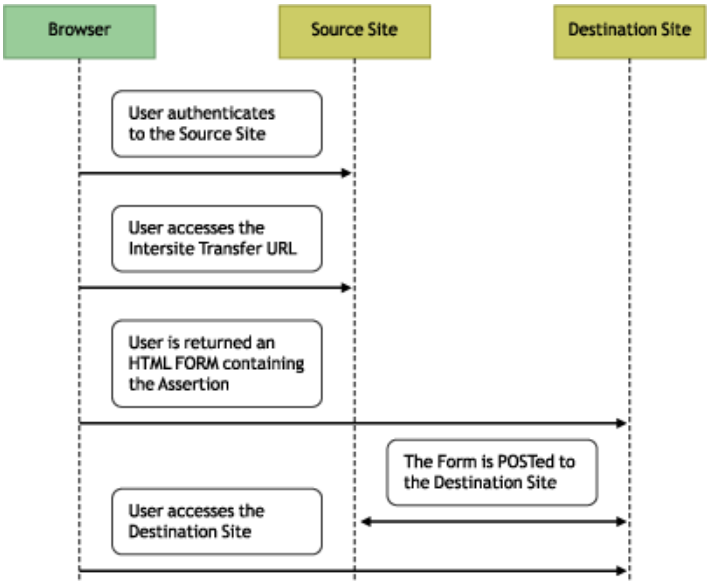
SAML Service URL

The service running at the Artifact Receiver URL must send the source site a SAMLP Request to obtain the Authentication assertion for the user (Step 4 of the Browser/Artifact Profile section). This implies that there is a SAML service running on the source site and listening for SAMLP Requests. This service is called the SAML Service and it is another URL running on the source site. This URL listens for incoming SOAP messages containing SAMLP Requests and attempts to fulfill them. In this example, the source site's SAML Service returns the assertion generated for the browser user by the Intersite Transfer service in Step 2 of the Browser/Artifact Profile section.

Browser POST Profile

The Browser POST profile is a simpler and somewhat less secure Web SSO. In this case, the SAML assertion is sent to the destination site in the form of an HTTP FORM POST. Figure 14 shows an activity diagram of the steps involved in this type of Web SSO:

Figure 14 Activity Diagram of Web SSO Steps



The steps involved in this transaction are:

1. The user authenticates to the source site.
2. The user accesses the Intersite Transfer URL.
3. The Intersite Transfer URL generates an assertion for the user and returns the assertion to the user in HTML form.
4. The HTML form containing the assertion is POSTed to the destination site's POST receiver URL.
5. The destination site redirects the user to the resource as indicated by the Target URL.

Intersite Transfer URL

The functionality of the Intersite Transfer URL is identical to the Browser/Artifact case, except that after the assertion is generated, rather than being stored waiting for retrieval by the destination site, it is Base 64-encoded and returned to the user in an HTML form. The Target URL is also returned in the same HTML form. The page can also contain Java script that automatically submits the form to the destination site POST Receiver URL.

Destination Site POST Receiver URL

The service running at this URL is nearly identical to that in the Browser/Artifact case. Instead of calling back to the source site SAML Service URL to obtain the assertion, the POST Receiver URL simply reads the assertion from the incoming request. The POST Receiver validates the assertion and re-directs the user to the Target URL.

The Browser/Artifact profile is generally preferred over the Browser/POST from the security standpoint because in the Browser/Artifact case, the assertion data is not required to pass through the user's browser.

Additional Transaction Profiles

There are a number of other possible ways of using SAML to solve security problems. Several examples discussed in the SAML specifications are:

- Back office transactions
- JAAS Authorization Service
- User Provisioning Service

These other SAML use cases have not yet received the exposure and attention that the Web SSO cases have. Web SSO is a good first step, but doesn't fully utilize all of the features and functionally of SAML. It is expected that as SAML technology becomes more widely deployed, these higher-level transactions will get more attention.

SAML Going Forward

SAML continues to be developed, and there is work currently being done to create the SAML 1.1 specification. This new specification will include a binding for SAML onto the WS-Security protocol. See the Oasis Web site (http://www.oasis-open.org) for details on the current state of the SAML specification and related SAML documents.

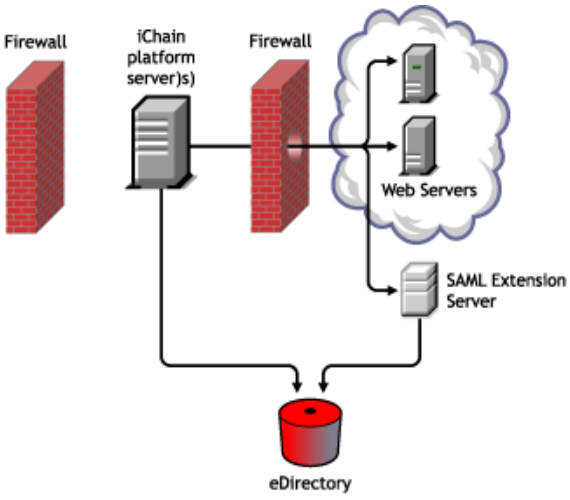
SAML Extension for Novell iChain

Novell iChain provides identity-based Web security services that control access to application and network resources. The SAML extension for Novell iChain adds SAML Web-based single sign-on and SAML attribute sharing to iChain's list of features. The SAML extension for Novell iChain gives iChain users the ability to generate SAML single sign-on assertions for outbound users, accept incoming SAML single sign-on assertions for incoming users, and respond to SAML attribute queries. The SAML extension for Novell iChain supports both the SAML Browser/POST and SAML Browser/Artifact Web single sign-on (SSO) profiles.

SAML Extension for Novell iChain Implementation

The SAML extension server runs as an HTTP server fronted by iChain. This means that the SAML service does not run on the iChain box itself, but runs on a separate Web server. This provides an additional level of security because the SAML service can be protected by a firewall and does not need to be accessible from the external network. The general topology of the SAML extension for Novell iChain is shown in Figure 15:

Figure 15 SAML Extension for Novell iChain Topology



All of the SAML processing occurs at the SAML extension server. This means that iChain-protected Web services can be enabled for SAML without deploying any additional software on the Web servers themselves. iChain communications with the SAML extension server by using a reserved URL prefix. The following two URLs are used by iChain to communication with the SAML extension server:

- **http(s)://*host*/cmd/ext:** All traffic that iChain receives with the /cmd/ext URI path prefix is sent to the SAML extension server. This traffic can be in SSL or cleartext format.
- **https://*host*/cmd/mutExt:** All traffic that iChain receives with the /cmd/mutExt URI path prefix is sent to the SAML extension server. This traffic must be received over SSL with client authentication.

The following sections will give a brief overview of how the interactions between iChain and the SAML extension for iChain server occur:

- “SAML Extension for iChain Assertion Generation” on page 24
- “SAML Extension for iChain Authentication” on page 25
- “SAML Extension for iChain Communication” on page 25

SAML Extension for iChain Assertion Generation

When users want to access a SAML partner, a SAML assertion must be generated for them. They are generally first sent to an Intersite Transfer URL resource. The SAML extension server provides two different types of Intersite Transfer URLs which are responsible for generating appropriate SAML assertions for users and forwarding those users to the proper resource at the partner site. The two URLs provided are:

- **http(s)://<host>/cmd/ext/samlext/saml/gen/post:** For outbound users using the SAML Browser/POST profile.

- **http(s)://<host>/cmd/ext/samlext/saml/gen/afct:** For outbound users using the SAML Browser/Artifact profile.

When iChain sees the /cmd/ext prefixes on the above URLs, it knows to forward the request to the SAML extension server. The SAML extension receives the request, generates the appropriate SAML assertions, and forwards the user to the partner site. More information on the use of the Intersite Transfer URLs can be found in Chapter 6, "Enabling Web Sites With SAML Single Sign-On Functionality," on page 101.

SAML Extension for iChain Authentication

Users can access an iChain-controlled resource using a SAML single sign-on assertion. This is done by accessing the iChain system using a specified SAML receiver URL. The SAML extension for iChain defines two SAML receiver URLs:

- **http(s)://<host>/cmd/ext/samlext/saml/auth/post:** For incoming users using the SAML Browser/POST profile.
- **http(s)://<host>/cmd/ext/samlext/saml/auth/afct:** For incoming users using the SAML Browser/Artifact profile.

When iChain sees the /cmd/ext prefixes on the above URLs, it knows to forward the request to the SAML extension server. The SAML extension receives the request, then obtains and validates the incoming user's SAML assertions. If the SAML extension server is able to validate the SAML assertions provided for the user, the user is authenticated to iChain and provided with the requested resource.

SAML Extension for iChain Communication

The SAML specification requires that in some cases, SAML servers must communicate directly. This is the case in the SAML Browser/Artifact profile. The receiving SAML server must request the SAML single sign-on assertion directly from the issuing site. This direct server-to-server communication is sometimes called the SAML back-channel. The SAML extension server listens for incoming SAML requests on the following URLs:

- **https://*host*/cmd/ext/samlext/saml/resp:** The SSL-only responder.
- **https://*host*/cmd/mutExt/samlext/saml/resp:** The SSL with client authentication responder.

When iChain sees the /cmd/ext or /cmd/mutExt prefixes on the above URLs, it knows to forward the request to the SAML extension server. The SAML extension server receives the SAML request, parses it, then returns the appropriate SAML response. The two types of SAML requests that are currently supported are SAML artifact queries and SAML attribute queries. SAML artifact queries are used to perform the SAML Browser/Artifact Web single sign-on profile. SAML attribute queries are used to provide user attribute sharing with SAML partner sites.

2 Installing the SAML Extension Software

This section provides instructions for installing the SAML extension for Novell® iChain® software and contains the following topics:

- SAML Extension Installation Prerequisites
- Product Components
- Installing SAML Extension Software
- Validating the Installation

SAML Extension Installation Prerequisites

The platforms Novell supports for installing the SAML extension for Novell iChain are:

- NetWare 6 with Support Pack 3
- Windows* 2000 with Service Pack 4
- Red Hat Linux 8

The SAML extension for Novell iChain is a self-contained installation and does not require licensed hardware to run. However, you must have the following requirements installed prior to installing the SAML extension software:

- Novell iChain 2.2
- Tomcat 4.0 on Linux* and Windows, or Tomcat 3.*x* on NetWare®
- JVM* 1.4*x*

See Appendix A, "Installing the Java Development Kit," on page 105 and Appendix B, "Installing Tomcat Light Edition," on page 107 for instructions on how to obtain these components.

IP Connectivity Requirements

The SAML server requires IP connectivity as follows:

- The server should be able to access the IP address of the LDAP server.
- The server should be able to resolve the DNS name of the accelerators that are using the SAML extension. (For example, accelerators that use http://*accelerator dns*/cmd/samlext...) This would include accelerators of all of the trusted sites.

Hardware Requirements

For hardware requirements, see the *iChain 2.2 Hardware Guide* (http://www.novell.com/products/ichain/hardware22.html).

For additional information and full system requirements for Novell iChain, refer to the Novell *iChain Administration Guide*, available at the Novell Documentation Web site (http://www.novell.com/documentation/lg/ichain22/index.html).

You can download Novell iChain at Novell Software Downloads (http://download.novell.com).

Product Components

The SAML extension for Novell iChain includes installing the following components:

- SAML extension server
- ConsoleOne® snap-ins
- SAML schema

Installing SAML Extension Software

The SAML extension for Novell iChain software should only be installed on compatible hardware. The installation is divided into four sections: an installation introduction, the server installation, ConsoleOne snap-ins installation, and schema installation.

For hardware requirements, see the *iChain 2.2 Hardware Guide* (http://www.novell.com/products/ichain/hardware22.html).

NOTE: The Novell-supported platforms for the SAML extension for Novell iChain software are Windows 2000, Red Hat Linux 8, and NetWare 6 with Support Pack 3. You should run the installation executable on the machine you are designating as the SAML server.

Also, this portion of the installation requires that the following environment variables are properly set:

CATALINA_HOME or TOMCAT_HOME: This value indicates where the Tomcat servlet engine has been installed. This value must be set manually. To create the setting, you need to create an environment variable named CATALINA_HOME that points to the Tomcat install directory. For example, if you installed Tomcat to c:\tomcat, then your CATALINA_HOME environment variable would be CATALINA_HOME=c:\tomcat.

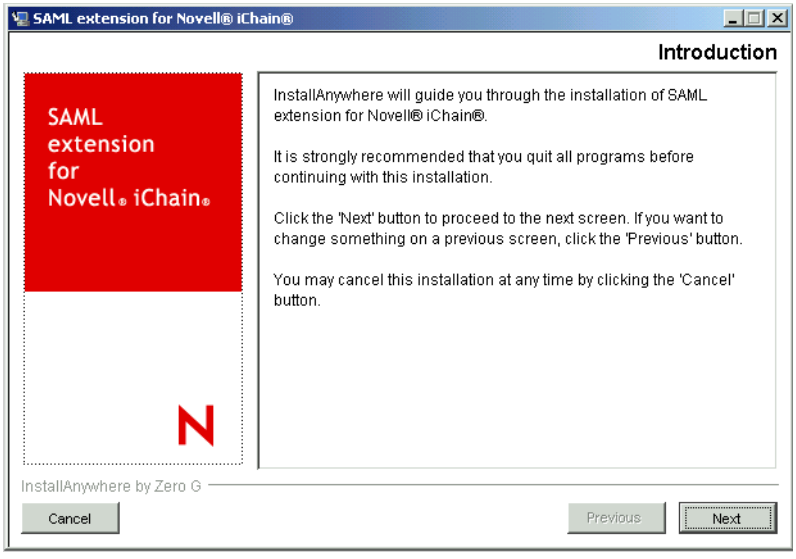
JAVA_HOME: This value indicates the location of Java on the system. If you installed Java using the Sun Java installer, this value was automatically set; however, it can also be set manually. To create the setting, you need to create an environment variable named JAVA_HOME that points to the Java home directory. For example, if you installed the Java Development Kit to c:\j2sdk1.4, your JAVA_HOME variable would be JAVA_HOME=c:\j2sdk1.4.

To install the SAML extension for Novell iChain:

1. At the Web download site, click the Web download link to automatically download the SAML extension executable.
2. Double-click the executable to launch the installer.

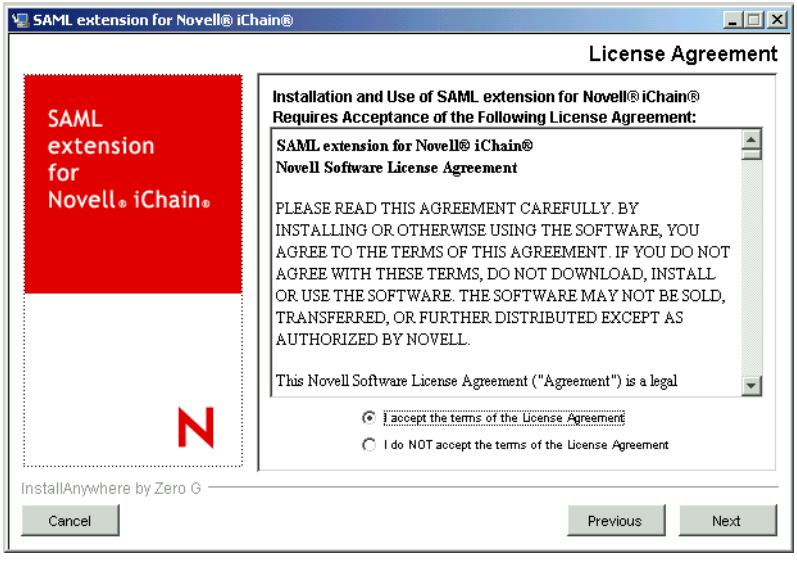
   The installation program will guide you through the installation process.

Figure 16 SAML Extension Introduction



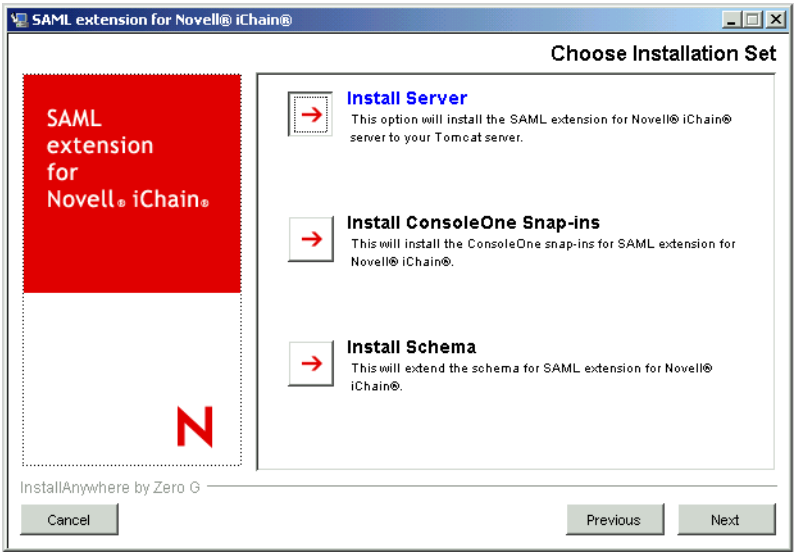
3 After you have read the introductory screen, click Next.

Figure 17 License Agreement



4 Accept the terms of the License Agreement, then click Next.

Figure 18 Choose Installation Set



5 Choose the installation component.

There are three different components included in the SAML extension installation. They are:

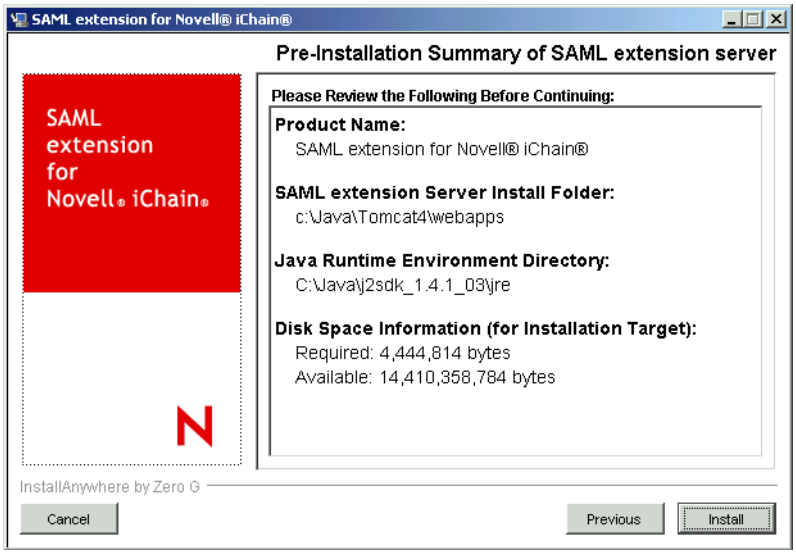
- **Install Server:** This option installs the SAML Extension for Novell iChain Web application. You should run this option on the machine you want to be the SAML extension server. This component requires that the system has Java Development Kit (JDK*) 1.4.1 or later and the Tomcat servlet engine installed.
- **Install ConsoleOne Snap-Ins:** This option installs the ConsoleOne snap-ins used to administer the SAML extension components.
- **Install Schema:** This option extends the eDirectory™ schema to include the SAML configuration object definitions.

The Install Server button is selected by default. These steps assume you will accept the default selection.

6 Click Next to begin the SAML extension server installation.

A pre-installation summary is displayed:

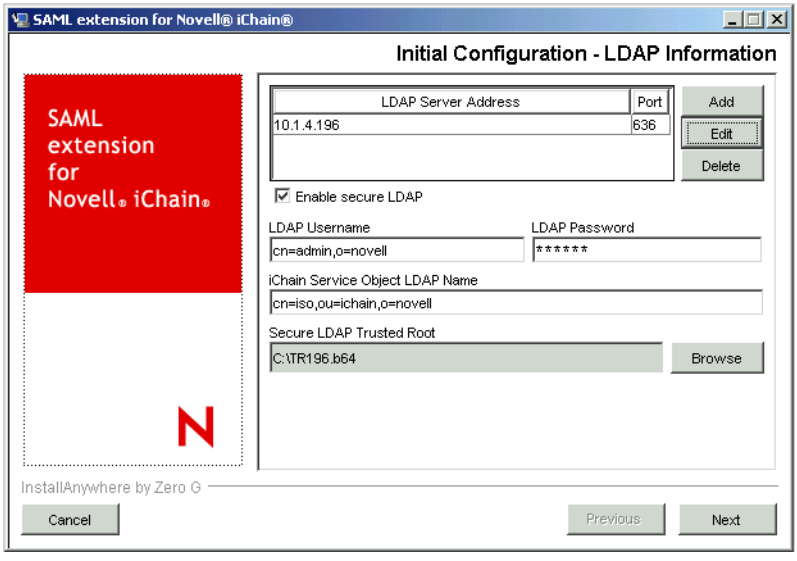
Figure 19 Pre-installation Summary of SAML Extension Server



7 Review the SAML extension server pre-installation summary, then click Install.

The SAML extension Web application is installed to the specific SAML extension server install folder. For example, Figure 19 shows the folder as c:\tomcat\webapps. The installer creates a folder named samlext in this directory. After the Web application files are installed, you are prompted for initial system configuration information, as shown in Figure 20.

Figure 20 Initial Configuration



8 Specify the initial configuration information.

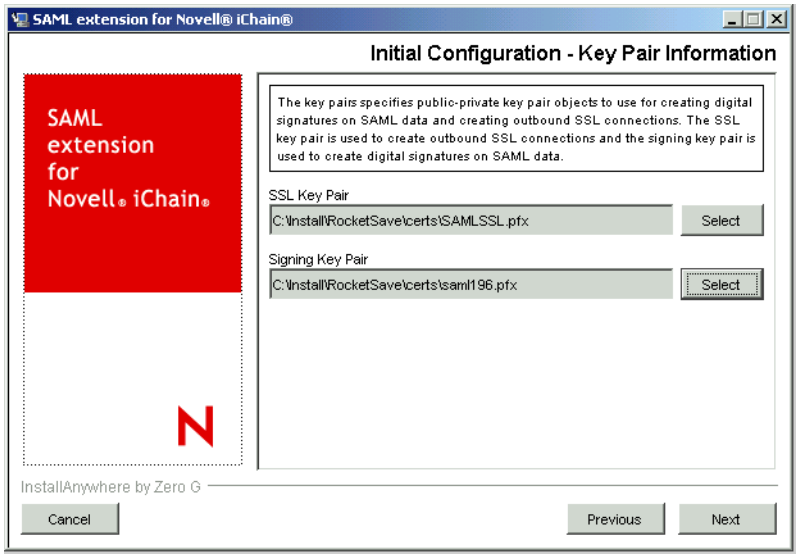
The following configuration entries (some of which are required) can be set:

- **LDAP Server Address:** The address of the LDAP servers that contain the configuration and users for the SAML extension server. These should be the same LDAP servers that contain the configuration and users for iChain. You must enter information in this field or the server does not function properly.

- **LDAP Username:** The user name of the proxy user that the SAML extension server uses to access the directory. We recommend that this user be the same as the one used by iChain.
- **LDAP Password:** The user password for the proxy user. You must enter information in this field or the server does not function properly.
- **iChain Service Object LDAP Name:** The name of the iChainServiceObject associated with this iChain installation. The SAML extension server uses this value to find its configuration objects in the directory. You must enter information in this field or the server does not function properly.

9 Click Next.

Figure 21 Initial Configuration: Key Pair Information



10 The following configuration entries can be set:

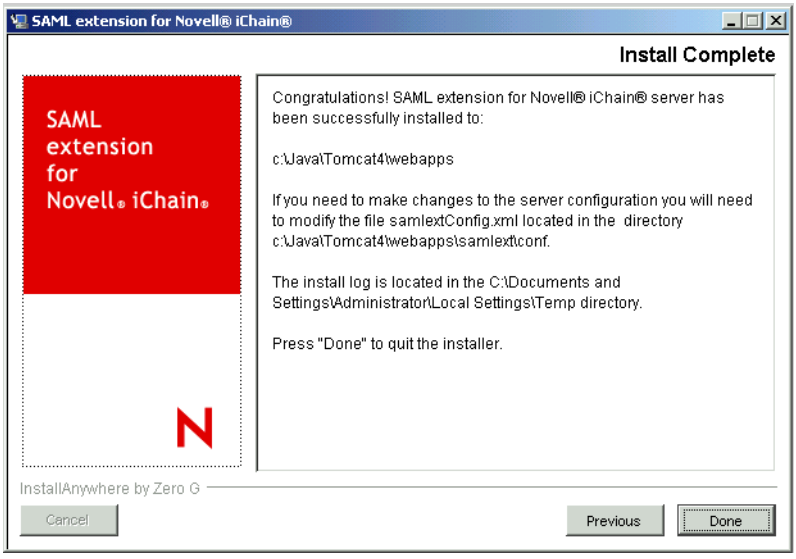
- **SSL Key Pair:** Allows the administrator to include a key pair in PKCS#12 or JKS format to be used for outbound SSL connections. You can choose to leave this field blank and configure it later.
- **Signing Key Pair:** Allows the administrator to include a key pair in PKCS#12 or JKS format to be used to sign SAML data. You can choose to leave this field blank and configure it later.

All of the settings you select are stored in a file on the server.

11 Click Next.

When the server installation has successfully completed, an Install Complete page appears.

Figure 22 SAML Extension Server Install Complete



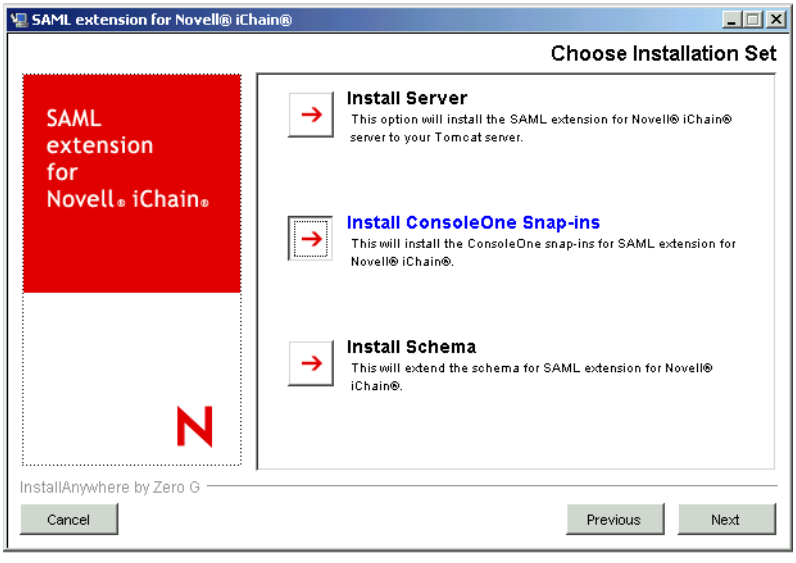
This dialog box shows the location of the file. For example, Figure 22 shows that the configuration information has been installed to c:\Java\Tomcat4\webapps\samlext\conf. You can make changes to this file if you need to change your configuration. For more information about this file, see Chapter 4, “Configuring the SAML Extension Server,” on page 67.

12 Click Done to exit the server installer.

You must launch the installer again to install the next component.

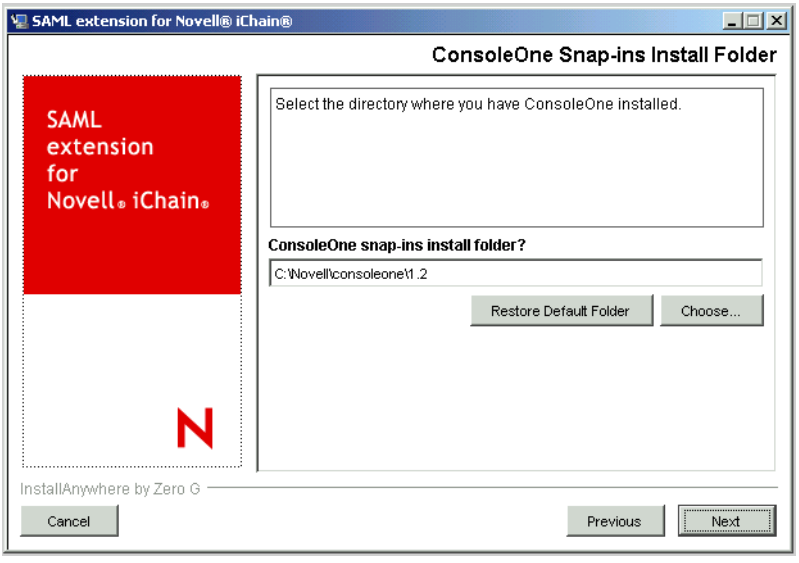
13 Launch the installer.

Figure 23 Choose Installation Set



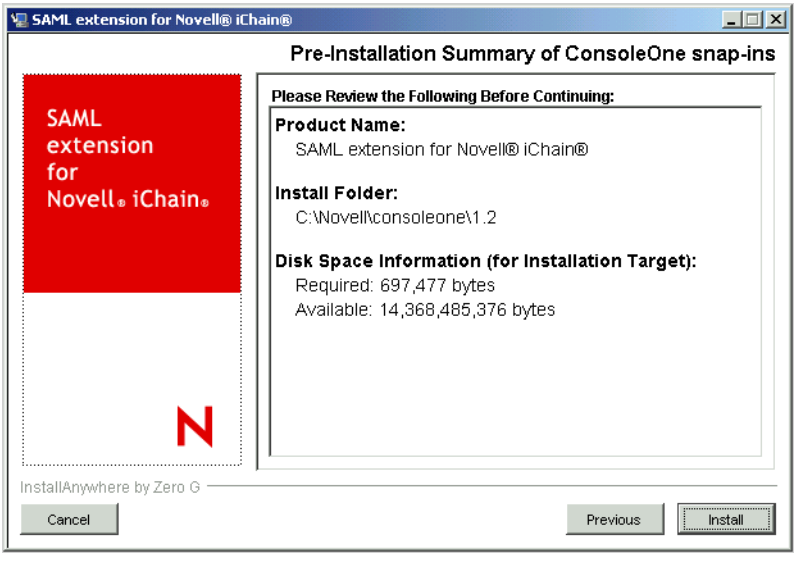
14 Click Install ConsoleOne Snap-ins, then click Next.

Figure 24 ConsoleOne Snap-ins Install Folder



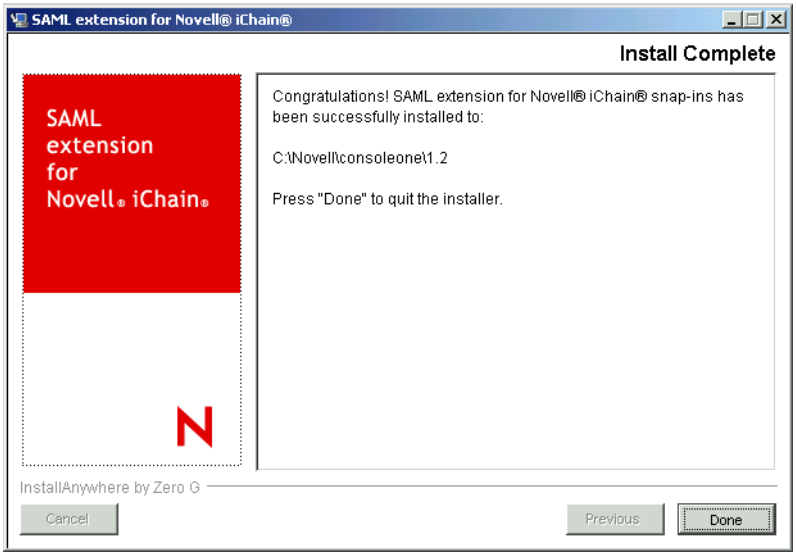
15 Select the directory on your machine where ConsoleOne is installed, then click Next.

Figure 25 Pre-Installation Summary of ConsoleOne Snap-ins



16 Review the ConsoleOne snap-ins pre-installation summary, then click Install.

Figure 26 ConsoleOne Snap-ins Install Complete

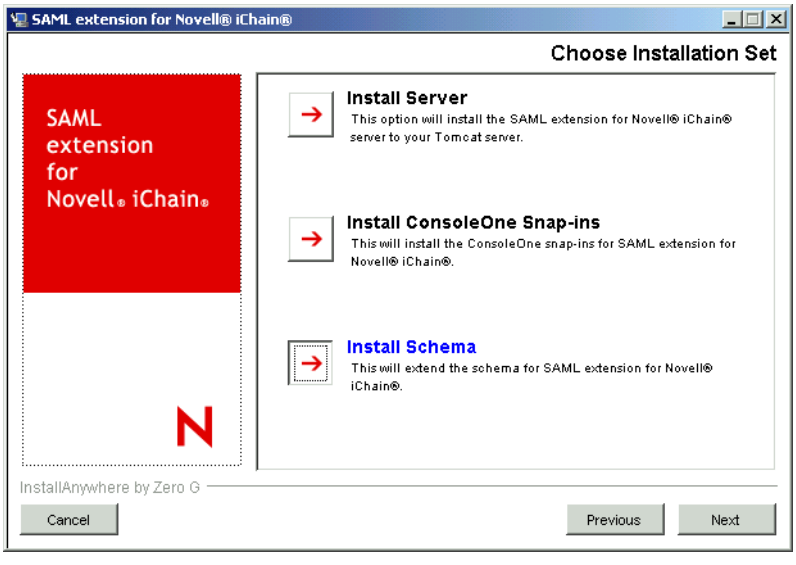


17 When the ConsoleOne snap-ins installation has successfully completed, an Install Complete screen appears. Click Done to exit the ConsoleOne snap-ins installer

You must launch the installer again to install the final component.

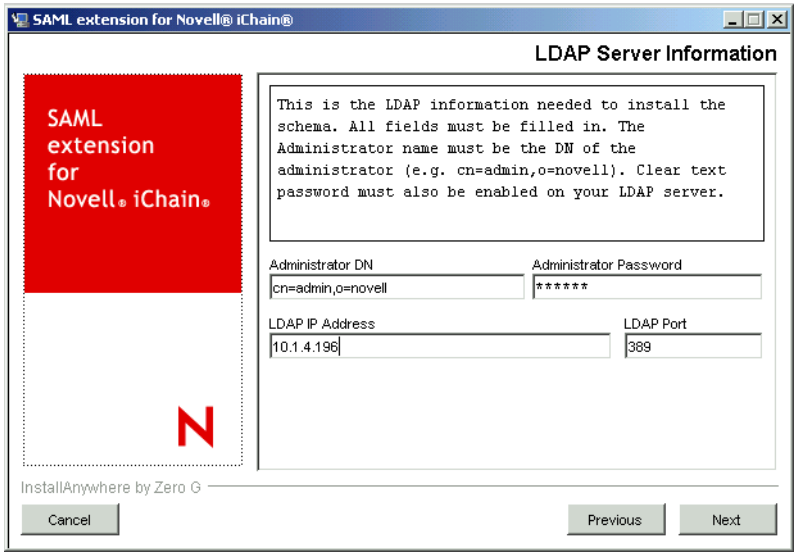
18 Launch the installer.

Figure 27 Choose Installation Set



19 Click Install Schema, then click Next.

Figure 28 LDAP Server Information

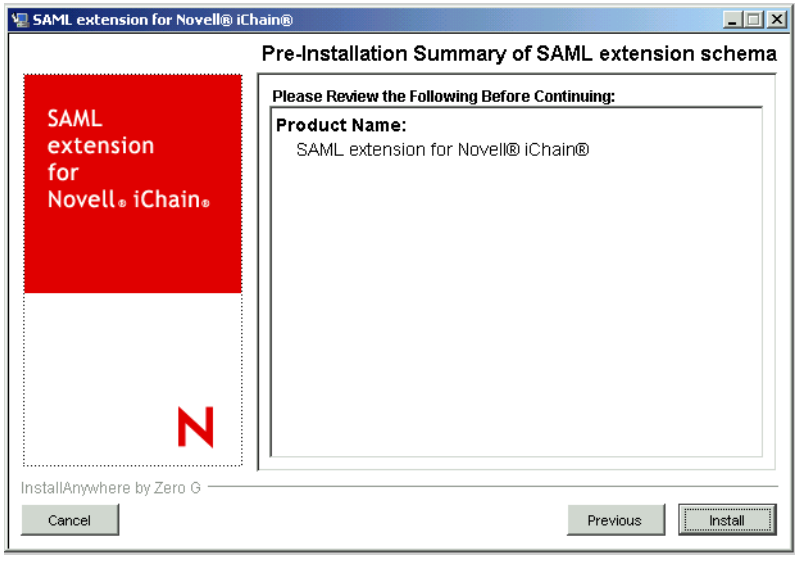


20 In order to install the schema, you must specify the Administrator DN, Administrator Password, LDAP IP Address, and LDAP Port information for your LDAP server. You must also enable clear text password on your LDAP server.

The LDAP server should be the same one you specified during the SAML extension server portion of the installation. For example, Figure 20 and Figure 28 show that the LDAP server address is 137.65.159.66.

21 Click Next.

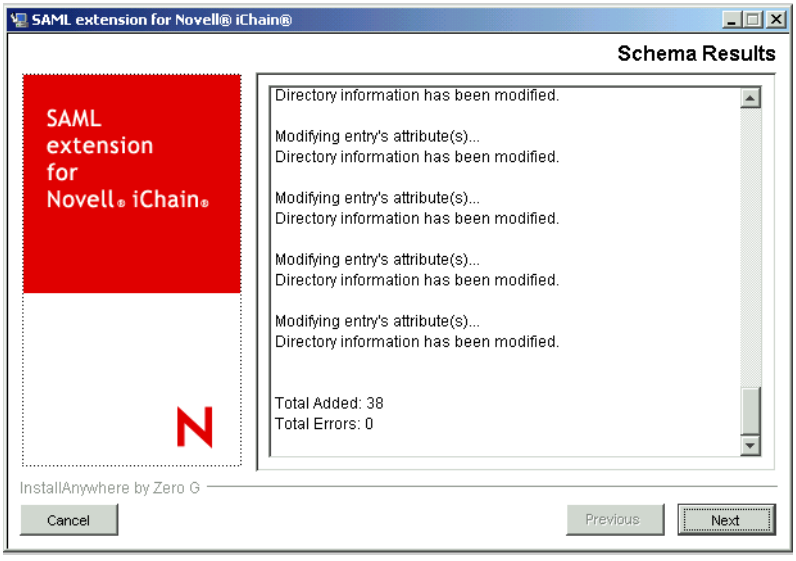
Figure 29 Pre-Installation Summary of SAML Extension Schema



22 Review the schema pre-installation summary, then click Install.

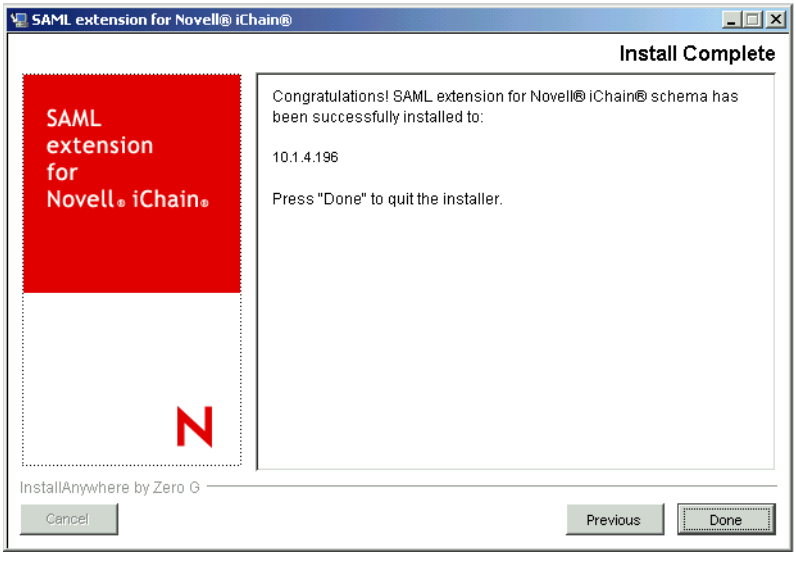
When the installation is complete, a schema results page is displayed.

Figure 30 Schema Results Summary



23 Review the schema results summary, then click Next.

Figure 31 Install Complete



24 When the schema installation has successfully completed, an Install Complete page appears. Click Done to exit the installation.

Validating the Installation

You can validate that the server components installed properly by opening ConsoleOne and browsing to the tree where you installed the SAML extension server schema extensions. You can create SAML extension server objects. See "Configuring SAML and ConsoleOne" in the SAML Extension for iChain Sample Site Setup Guide for details.

When you have completed creating the SAML extension server configuration in the directory, you can start the Tomcat servlet engine. Because the SAML extension Web application is installed into the Tomcat webapps directory, it should be automatically deployed. You can validate this by entering the following URL in a browser: http://*address of SAML Extension Server*/samlext/status. You should see a page that indicates the status of the SAML extension server. If you get a 404 error page, the application was not deployed.

Troubleshooting the Installation

This section addresses how to resolve common issues with your SAML extension server installation. The following topics are addressed:

- java.lang.OutOfMemory Error
- Performance Tuning Apache and Tomcat

java.lang.OutOfMemory Error

The default Tomcat Java memory setting is too low for the SAML extension server to run consistently. This out-of-memory error is very common, and to resolve it, you need to manually increase your memory. Java only starts with 16MB allocated and uses up to a maximum of 64MB, no matter how much memory your server has. You can increase these values using command line parameters when Tomcat is started by Java.

Refreshing the browser page occasionally bypasses this error; however, if you cannot resolve the problem by refreshing the browser page, see the Novell Technical Information Document (http://support.novell.com/cgi-bin/search/searchtid.cgi?/10068408.htm) for instructions on how to increase your Java memory.

Performance Tuning Apache and Tomcat

If you observe that the SAML extension server is showing poor performance, you might need to fine-tune your Apache Tomcat server. One method is to fine-tune the number of request processing threads.

At Tomcat server startup time, the Connector (as defined in the Tomcat server's server.xml file) creates a number of request processing threads (based on the value configured for the minProcessors attribute). Each incoming request requires a thread for the duration of that request. If more simultaneous requests are received than can be handled by the currently available request processing threads, additional threads are created, up to the configured maximum (the value of the maxProcessors attribute). If still more simultaneous requests are received, they are stacked up inside the server socket created by the Connector, up to the configured maximum (the value of the acceptCount attribute. Any further simultaneous requests will receive “connection refused” errors, until resources are available to process them

To increase the number of request processing threads, you can edit the Tomcat's server.xml (nwserver.xml on NW6 Tomcat33) and change the maxProcessors value of the Connector Section to suit your production environment. (If not specified, this attribute is set to 20.) The server.xml file is found in the *CATALINA_HOME*\conf (or *TOMCAT_HOME*\conf) folder and is as shown below:

```
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080 -->
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
     port="8080" minProcessors="5" maxProcessors="75"
     enableLookups="true" redirectPort="8443"
     acceptCount="100" debug="0" connectionTimeout="20000"
     useURIValidationHack="false" disableUploadTimeout="true" />
```

3 Configuring the SAML Extension

This section discusses the types of directory objects associated with the SAML extension for Novell® iChain® and how to configure SAML extension. The following topics are included:

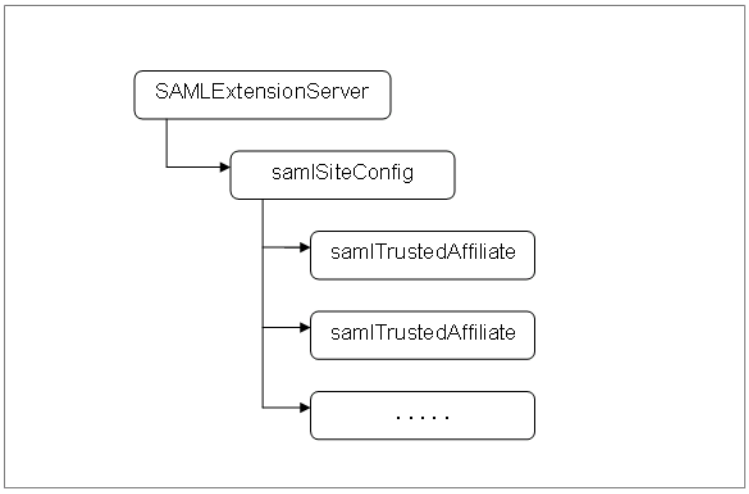
SAML Extension Directory Objects

There are three different types of directory objects associated with the SAML extension for Novell iChain:

- **SAMLExtensionServer:** Contains configuration information that allows the iChain server(s) to communicate with the SAML extension server.
- **samlSiteConfig:** Contains the top-level SAML configuration for the system, and contains attributes that define this SAML site. The samlSiteConfig object is contained by the SAMLExtensionServer object.
- **samlTrustedAffiliate:** Contains information about this site's relationship with a SAML partner site. It contains all of the settings that allow this site to communicate and trust the partner site. The samlTrustedAffiliate object is contained by the samlSiteConfig object.

Figure 32 shows the directory layout of the SAML extension for Novell iChain directory objects:

Figure 32 SAML Extension Directory Objects



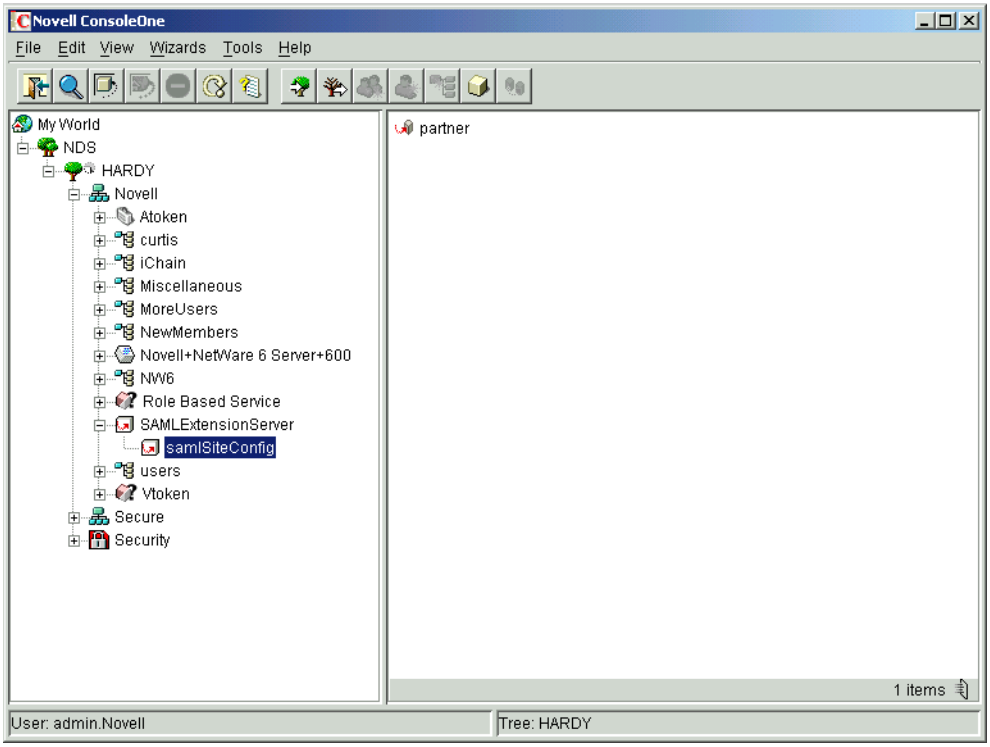
SAML Trust Relationships

To illustrate how the SAML extension relationship works, consider the following example: There are two sites that want to create a SAML relationship: Novell and PartnerCorp. Both Novell and PartnerCorp need to have some sort of SAML configuration. For the purposes of this example, it is assumed that both entities are using the SAML extension for Novell iChain. The following configurations are needed:

1. Novell SAML configuration
   a. Create a samlSiteConfig to represent MYSELF
      i. Create a Site ID: www.novell.com
      ii. Create a Source ID = XYZ
   b. Create a samlTrustedAffiliate to represent my relationship with PartnerCorp
      i. Use the Site ID provided from PartnerCorp: www.partnercorp.com
      ii. Use the Source ID provided from PartnerCorp: PDQ
2. Partner SAML configuration
   a. Create a samlSiteConfig to represent MYSELF
      i. Create a Site ID: www.partnercorp.com
      ii. Create a Source ID: PDQ
   b. Create a samlTrustedAffiliate to represent my relationship with Novell
      i. Use the Site ID provided from PartnerCorp: www.novell.com
      ii. Use the Source ID provided from Novell: XYZ

Figure 33 shows the directory object layout of each of these configurations. The left side of this window shows the configuration for Novell and right side shows the configuration for PartnerCorp.

Figure 33 Directory Object Layout



With this configuration, when PartnerCorp receives a SAML assertion issued by Novell, PartnerCorp can identify the assertion with its samlTrustedAffiliate entry for Novell because of the matching Site ID to Issuer value. Also, when PartnerCorp receives a SAML Artifact from Novell (XYZ), it can associate that artifact with Novell because of the matching Source ID value.

Much more than the Site ID and Source ID must be shared in order to create a SAML trust relationship. At the current time there is no standard way of sharing this configuration information. There is work going on in the SAML standards body to create a common metadata format that SAML partner sites could exchange to automatically create these trust relationships. However, until that work is complete, the process must be done by hand in an out-of-band communication between SAML system administrators. Typically, the necessary information to create a SAML trust relationship includes the following:

- **Site ID:** The SAML Site ID for the partner.
- **Source ID:** The SAML Source ID for the partner.
- **SOAP Endpoint URL:** Where the partner receives SAML SOAP messages.
- **Artifact Receiver URL:** Where the partner receives incoming SAML Artifacts.
- **POST Receiver URL:** Where the partner receives incoming SAML POST data.
- **Signing Public Key Certificate:** The public key certificate used to sign SAML data.
- **SSL Server Certificate:** Allows us to trust their SAML server to make client requests.
- **SSL Client Certificate:** Allows us to trust their SAML client to make server requests.
- **Requested User Attributes:** What does the other site need to know about my users?
- **Audiences:** What audience restriction conditions will be in placed on SAML assertions?

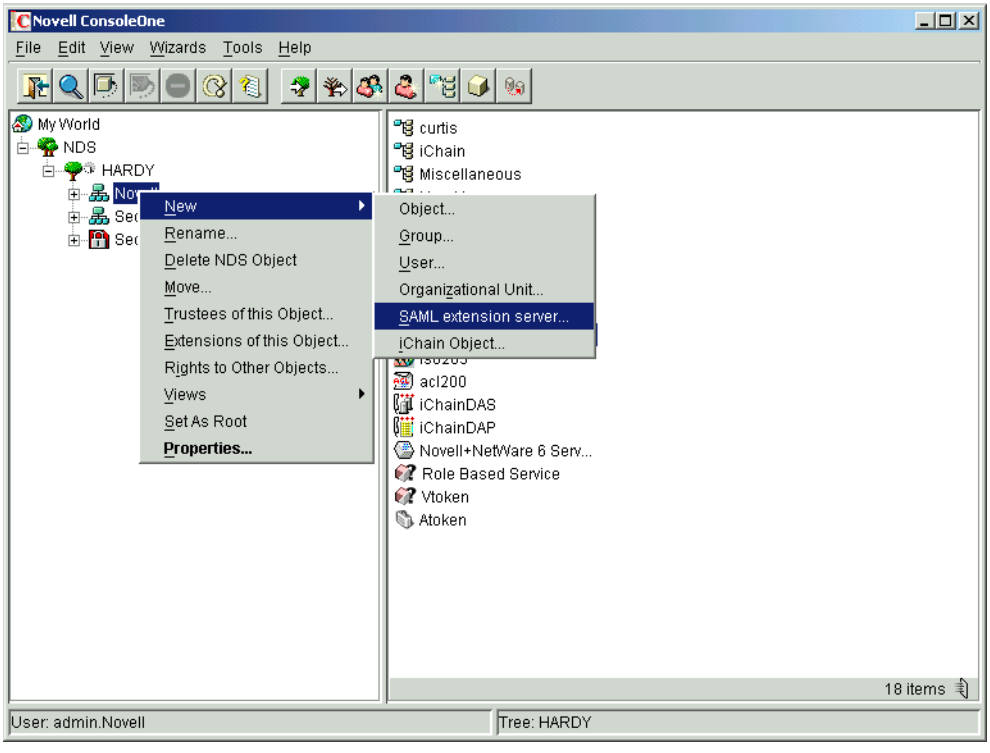
These settings could be shared between the sites using e-mail, and some could even be negotiated in telephone conversations.

The following sections deal with the objects in the directory that are used to configure the SAML system and to define these SAML trust relationships.

Creating and Configuring the Identity Provider Site Object

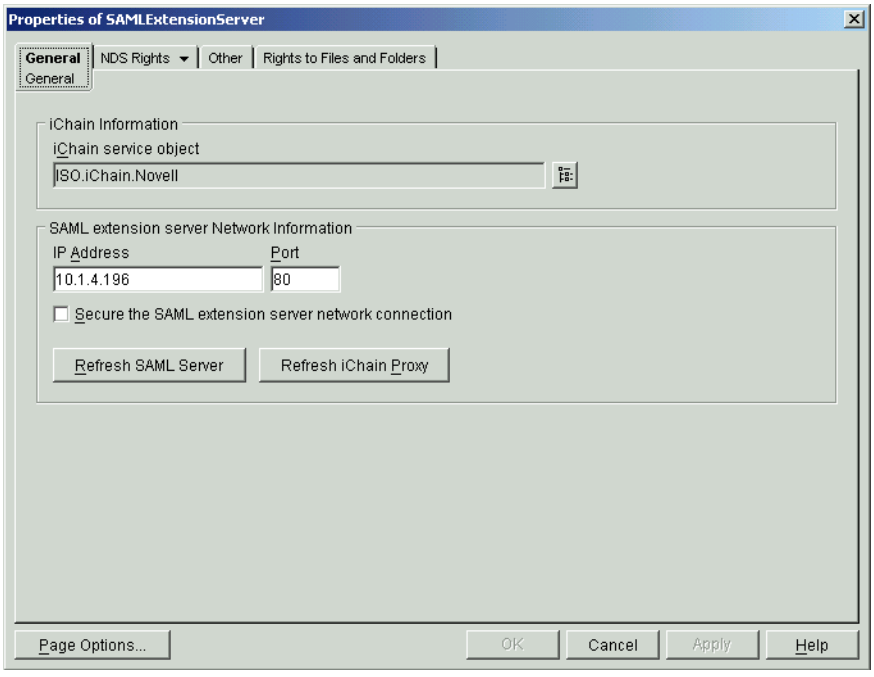
The SAMLExtensionServer object provides basic information to iChain that allows iChain to communicate with the SAML extension server. You create it by right-clicking on the organization or organizational unit you want to create it in, then clicking New > SAML extension server..., as shown in Figure 34:

Figure 34 Creating a New Provider Site



After you create the SAMLExtensionServer object, right-click it and select Properties to display the Properties page. For the SAMLExtensionServer only, this page contains a single custom tab, called the General tab, which displays the page shown in Figure 35:

Figure 35 General Page



The General page contains settings that allow the iChain servers to communicate with the SAML extension for iChain server. iChain communicates with the SAML extension similarly to how it communicates with back-end Web servers. When iChain receives traffic with the URL prefix of /cmd/ext or /cmd/mutExt, the HTTP request is sent to the SAML extension server specified here (rather than to the accelerator Web server). In order to do this, iChain must know the IP address and HTTP listening PORT that the SAML extension server is running, just as it needs to know this information for its back-end Web servers. The following are the properties available on this tab:

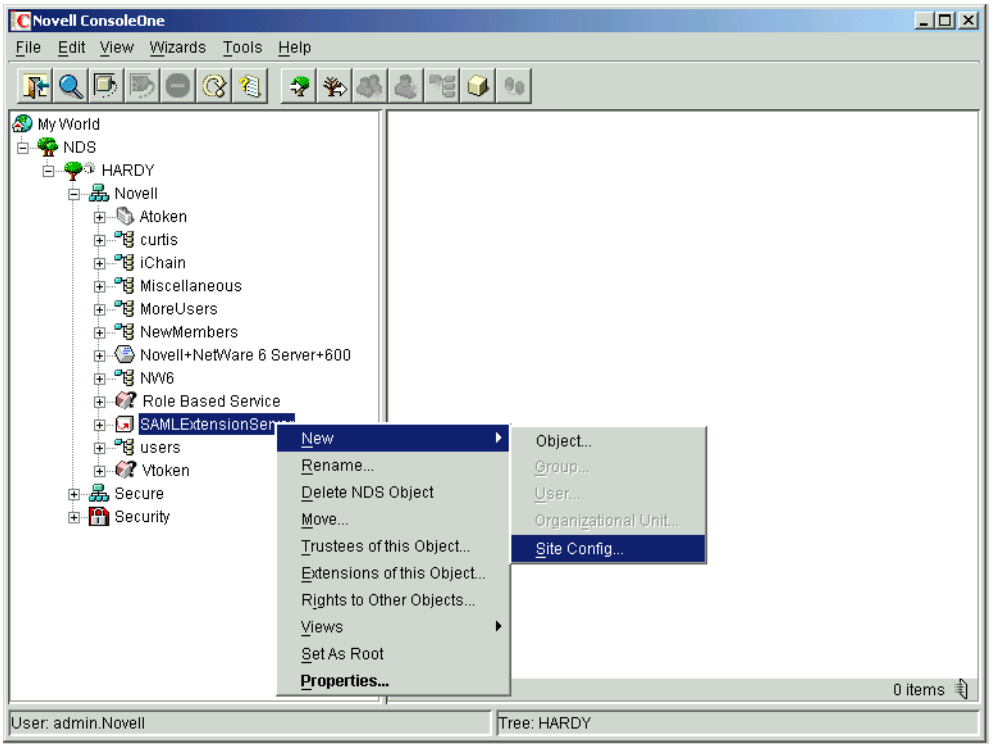
- **IP Address:** The IP address of the SAML extension server.
- **Port:** The port on which the SAML extension server host HTTP service is running.
- **Secure the SAML Extension Server Network Connection:** Selecting this option causes iChain to make SSL connections with the SAML extension server. In order do this, you must import the public key or trusted root certificate corresponding to the SAML extension server's SSL server certificate into the iChainServiceObject's trusted roots container.

  When enabling SSL between iChain and the SAML server, any accelerator handling incoming SAML Posts and Artifacts needs to have Secure Exchange enabled between the browser and the iChain proxy. If Secure Exchange is not enabled, the user receives a 504 Gateway timeout error from iChain.
- **Refresh Server:** Causes the SAML extension to reread the configuration information from the directory.

Creating and Configuring the SAMLSiteConfig Object

The samlSiteConfig object contains the configuration information that defines your site as a SAML service provider. You create it by right-clicking the SAMLExtensionServer, then selecting New > Site Config, as shown in Figure 36:

Figure 36 Creating a SAMLSiteConfig Object

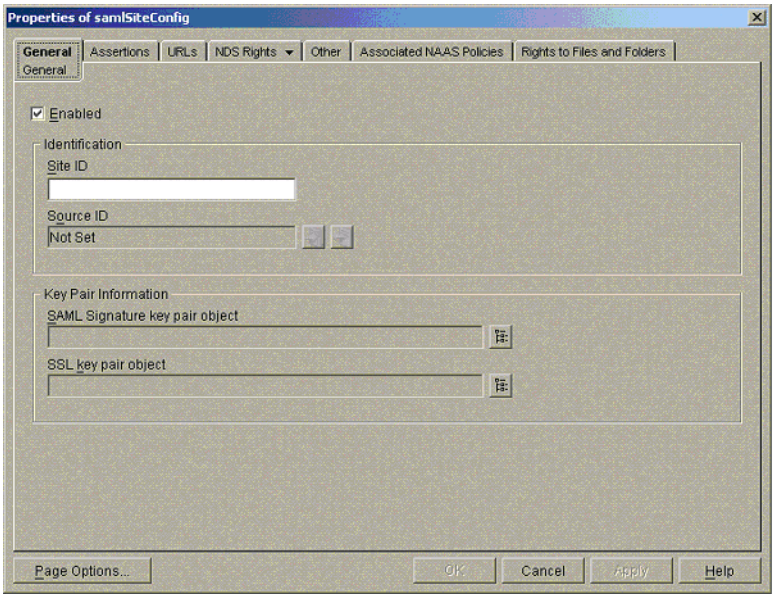


The samlSiteConfig object contains the top-level SAML configuration information that is used to identify this site to other SAML sites. The samlSiteConfig property page has three main tabs: General, Assertions, and URLs.

General

The General page contains identification information that is used to identify your site to SAML partner sites. When generating assertions, the SAML issuer and Source ID are taken from the settings on this page. Figure 37 shows the samlSiteConfig's General page. The available settings are described below.

Figure 37 samlSiteConfig: General Page

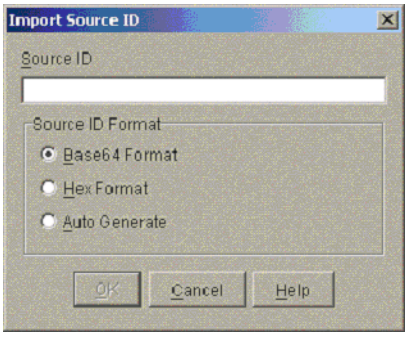


Enabled: Globally enables or disables the SAML service.

Identification - Site ID: Contains the SAML Site ID of this site. This value is used as the SAML Issuer ID on generated SAML assertions, and it must be shared with SAML partner sites. The Site ID can be any string value, but it is a good idea to use a value that describes the company. For example, if you were configuring the SAML system for a company named Novell, you could use Novell or www.novell.com.

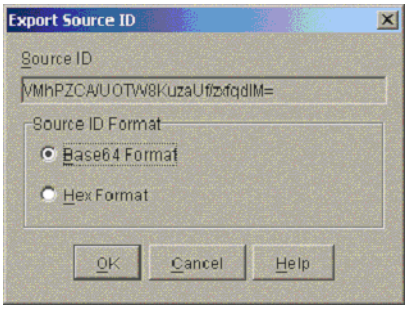
Identification - Source ID: A 20-byte value that is used as part of the Browser/Artifact profile. It allows the receiving site to determine the source of received SAML Artifacts. In most cases, the Source ID can be auto-generated using a SHA-1 hash of the Site ID. A utility is provided to either import or auto-generate the Source ID. Clicking the button directly to the right of the Source ID text field invokes the Import Source ID dialog box as shown in Figure 38:

Figure 38 samlSiteConfig: Import Source ID



The Source ID can be imported from a Base64 or HEX-encoded string, or it can be auto-generated using the SHA-1 hash of the Site ID. Just like the Site ID, this value must be shared with your SAML partner sites. The Source ID can be sent to partner sites in encoded Base 64 or HEX format. The button directly to the right of the Import Source ID button invokes the Export Source ID dialog box, as shown in Figure 39:

Figure 39 samlSiteConfig: Export Source ID



You can copy the Source ID value from the text field in either Base64 or HEX format and send it to SAML partner sites.

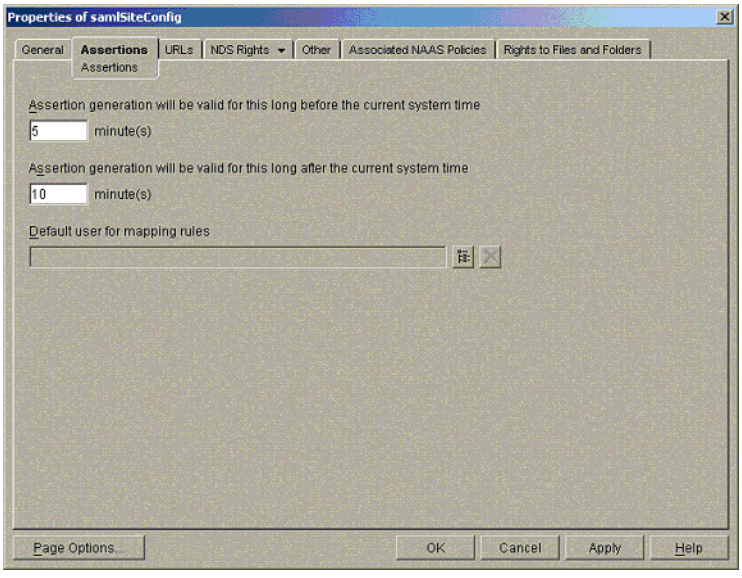
Key Pair Information: SAML Signature Key Pair Object: Allows the association between an NDSPKI: Key Material Object and the SAML service. This Key Pair object should be used to create digital signatures on generated SAML assertions. This value is for informational and book-keeping purposes only. In order to actually use the key pair to create digital signatures, you must export it from the directory in PKCS#12 format and copy it to the SAML Extension server. See Chapter 5, “SAML Security Considerations,” on page 71 for details on how to do this.

Key Pair Information: SSL Key Pair Object: Allows association between an NDSPKI: Key Material Object and the SAML service. This Key Pair object should be used to create mutually authenticated SSL connections between this site and partner sites. This key is used as the client key in mutually authenticated SSL connections. This value is for informational and book-keeping purposes only. In order to actually use the key pair to create mutually authenticated SSL connections, you must export it from the directory in PKCS#12 format and copy it to the SAML extension server. See Chapter 5, “SAML Security Considerations,” on page 71 for details on how to do this.

Assertions

The Assertions page defines the default SAML assertion generation behavior for the SAML service. Figure 40 shows the samlSiteConfig’s Assertions page. The available settings are described below.

Figure 40 samlSiteConfig: Assertions Page



Assertion Generation Will Be Valid for This Long before the Current System Time: SAML assertions contain a time stamp value that determines the time window during which they should be considered valid. SAML partner sites usually do not have system clocks that exactly match. A clock "pre-skew" is added to make up for this clock difference between partners. The default value is 5 minutes, but you can set this value depending upon how much difference you expect between your clock and your partner's system clock.

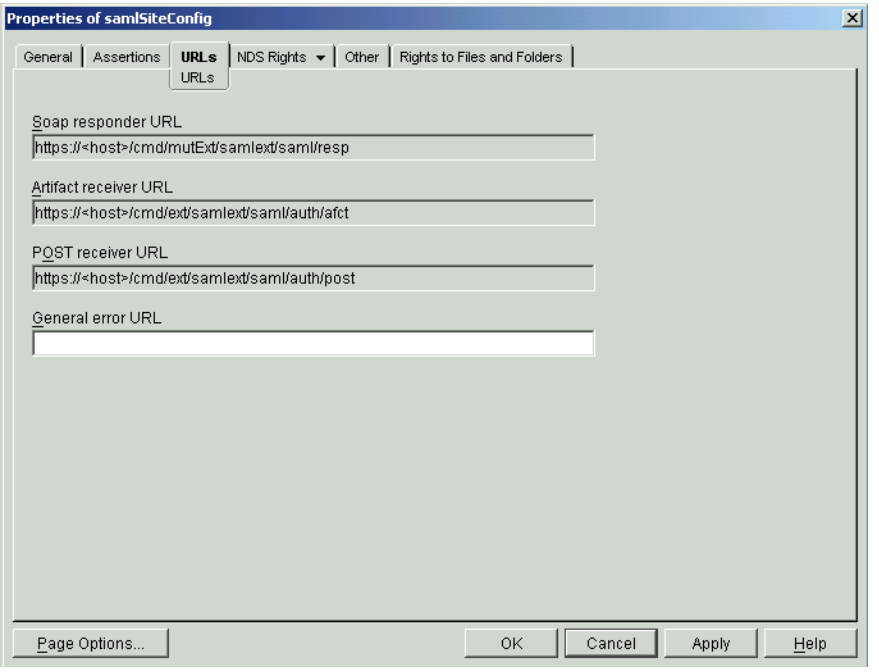
Assertion Generation Will Be Valid for This Long After the Current System Time: Determines how long after the current time the SAML assertion is considered valid. The value is generally larger than the previous value, to take into account network latency. The reason the time stamp condition is added to the SAML assertion is to prevent re-play or stolen SAML assertion attacks. In general, these validity period values should be kept as small as possible without introducing time skew errors for users.

Default User for Mapping Rules: Allows the designation of a default user to which incoming SAML users will be mapped in the event that no user mapping rules are defined, or all of the user mapping rules have failed. This value can be left blank, but users who are unable to be mapped will not be able to access the site and will receive an error message. See "User Mapping" on page 51 for more details.

URLs

The URLs page is mainly provided as a convenience for the SAML administrator to facilitate the sharing of configuration information. Three of the four URLs listed on the page are read-only because they are defined by the SAML extension server. Figure 41 shows the samlSiteConfig's URLs page. The available settings are described below.

Figure 41 samlSiteConfig: URLs Page



The first three (dimmed) URLs listed are for informational purposes only. They are provided on this page to facilitate the sharing of SAML configuration between this site and SAML partner sites. In order to create a SAML relationship with another site, three URLs must be known: the SOAP Responder URL, Artifact Receiver URL, and POST Receiver URL. For SAML implementations using the SAML extension for Novell iChain, the pattern for these three URLs will always be the same. For example, if you were SAML-enabling a site with host domain www.sample.com, the URLs would be:

SOAP: https://www.sample.com/cmd/mutExt/samlext/saml/resp

Artifact: https://www.sample.com/cmd/ext/samlext/saml/auth/afct

POST: https://www.sample.com/cmd/ext/samlext/saml/auth/post

When creating SAML relationships with other sites, you need to give them these URLs so that they know how to communicate with you. The URLs are listed here to aid the SAML administrator in getting these URLs to SAML partner sites.

General Error URL: Provides an error page URL that is displayed to the user if an error occurs while attempting to perform a SAML operation. The child samlTrustedAffiliate sites provide additional error URLs that can provide more fine-grained error handling. This general error URL is only used if the specified samlTrustedAffiliate has no error URLs defined, or the error occurred before the proper samlTrustedAffiliate configuration object could be found.

SAML Trusted Affiliate Object

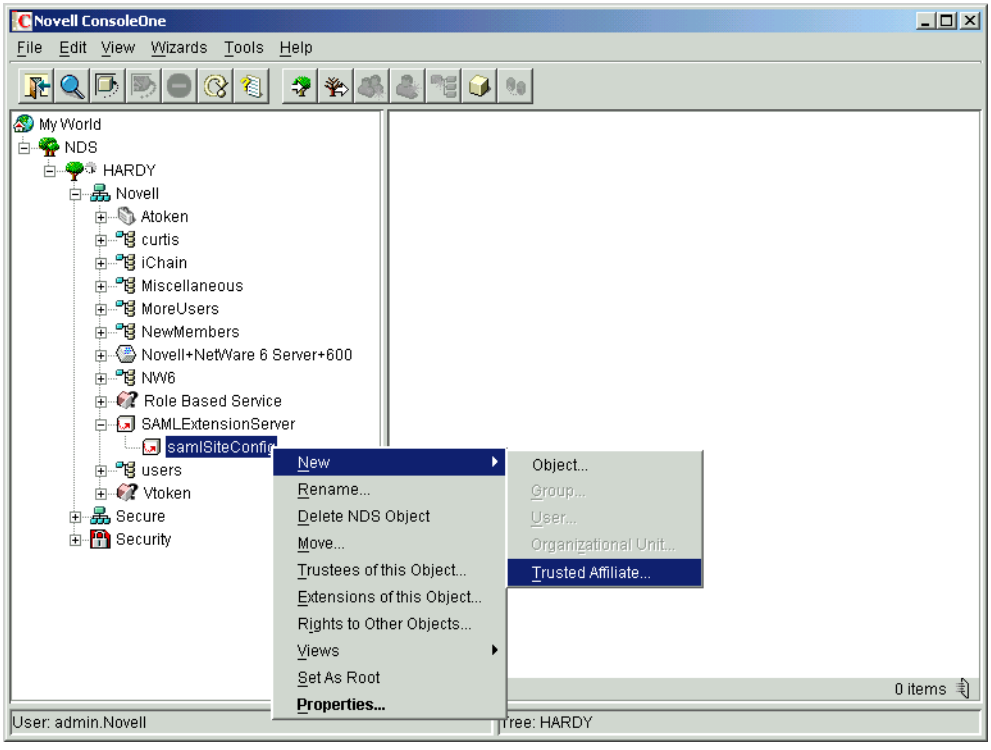
The samlTrustedAffiliate object defines the SAML trust relationship between your site (as defined in the samlSiteConfig object) and a SAML partner. The relationship between two SAML partner sites has two halves: settings that deal with incoming users from the partner site and settings that define how outgoing users going to the partner site are handled.

- Incoming SAML settings define how incoming SAML users from this Trusted Affiliate site are handled.
- Outgoing SAML settings define how to generate SAML assertions for users going to this Trusted Affiliate site.

(Where there could be confusion, this document clarifies which settings apply to which half of the relationship.)

To create a samlTrustedAffiliate object, right-click a samlSiteConfig object, then select New > Trusted Affiliate. See Figure 42. It is good idea to name the samlTrustedAffiliate object something similar to the actual SAML Site ID of the SAML partner site.

Figure 42 Creating a SAML Trusted Affiliate Object

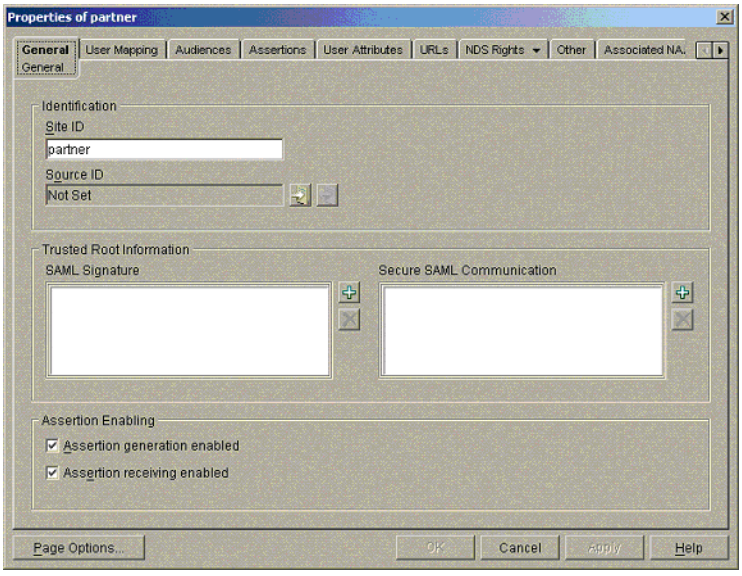


Open the properties page of a samlTrustedAffiliate by right-clicking it and selecting the properties menu. Each of the property pages are described below:

General

Figure 43 shows the samlTrustedAffiliate General page. This page contains the identification settings for this Trusted Affiliate object. This page is similar to the General page for the samlSiteConfig object. The settings for this page are described below.

Figure 43 samlTrustedAffiliate: General Page



Site ID: Contains the SAML Site ID for this Trusted Affiliate. This value is used to identify the Trusted Affiliate when performing SAML operations. It is used to determine the source of SAML assertions generated by this partner site.

Source ID: Contains the SAML Source ID for this Trusted Affiliate, and is used as part of the SAML Browser/Artifact profile. In the Browser/Artifact profile. an incoming user is presented to the SAML service with a SAML artifact. The SAML artifact contains two critical pieces of information:

- **Assertion Handle:** A reference to the SAML assertion generated for the user that the receiving site must request from the referring site.
- **Source ID:** A 20-byte value that uniquely identifies the referring site. The receiving site uses this value to know who to request the assertion from.

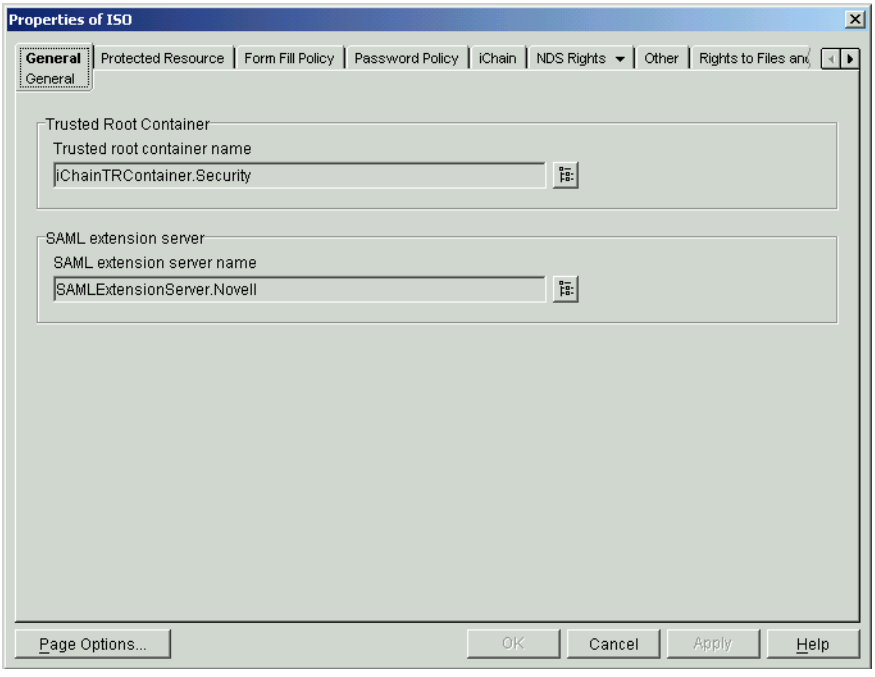
  The Source ID for the Trusted Affiliate can be imported or auto-generated, as shown on the samlSiteConfig properties page. In general, this value is received from the partner site in either HEX or Base64 format and is imported.

SAML Signature: Located under Trusted Root Information. This partner site might include a digital signature on SAML data it generates. You must have a public key certificate associated with the key used to generate the digital signature in order to validate it. This setting allows the administrator to specify the certificates that are to be used to validate digitally signed SAML information from this partner site. These certificates must have been imported into the directory using the Novell Certificate Server utilities. See Chapter 5, “SAML Security Considerations,” on page 71 for details on the SAML security settings.

Secure SAML Communication: Located under Trusted Root Information. Direct communication between the SAML extension server and the Trusted Affiliates is generally done over mutually authenticated SSL. In order to create these SSL connections, you must trust the server certificate associated with your partner's SAML service. The trusted roots listed here are used to create these connections. These certificates must have been imported into the directory using Novell Certificate Server utilities. Since incoming SSL requests pass through the iChain server, the SSL communication trusted roots must be imported into the same Trusted Root

container as that specified in the iChainServiceObject. This setting is found in the iChainServiceObject property page on the General page as shown below:

Figure 44 iChainServiceObject Property Page



Assertion Generation Enabled: Indicates whether you allow assertions to be generated for this Trusted Affiliate. If you set this to Off, you cannot send users from your site to this Trusted Affiliate site.

Assertion Receiving Enabled: Indicates whether you will allow incoming users from this Trusted Affiliate. If you set this to Off, incoming users from this Trusted Affiliate cannot access your site.

User Mapping

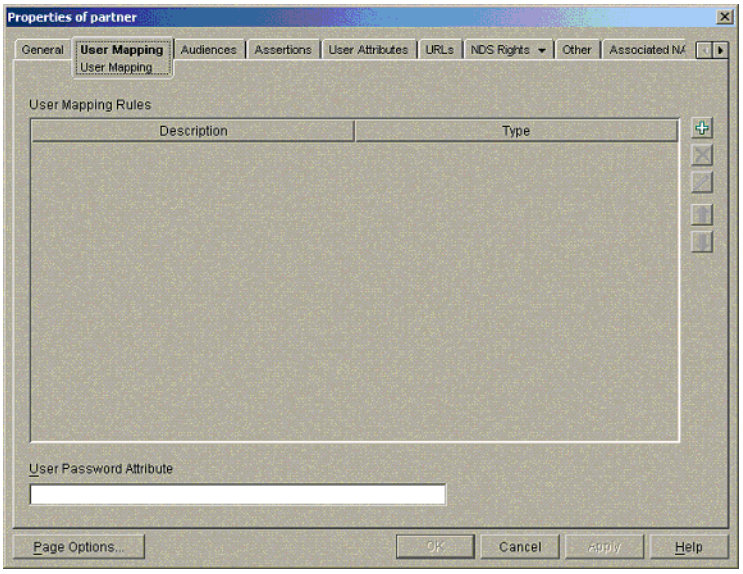
The User Mapping page deals exclusively with the incoming half of the relationship. The user mapping rules define how incoming SAML users from this Trusted Affiliate are mapped to identities at your site. There are two different user mapping rule types:

- **Dynamic:** Performs an LDAP search at a specified search base in the directory. Any attribute on the incoming assertion can be used as a search parameter. The first matching user object that is found matching the specified criteria are used as the user identity at this site. This type of rule is good for 1-to-1 mappings where this site and the partner site have the same set of users.
- **Static:** Performs a comparison operation of a static value against a value in the incoming SAML assertion. If the comparison operation evaluates to TRUE, then a statically defined user mapping takes place. This type of rule works well for many-to-one mappings where users are mapped to roles rather than their individual identity.

The user mapping rules are stored in an ordered list. This means they are evaluated in order until a mapping is found. If no mapping rules exist or no mapping is found, the default user mapping as defined in the samlSiteConfig object is applied. If there is no default user mapping, the incoming user cannot access your site.

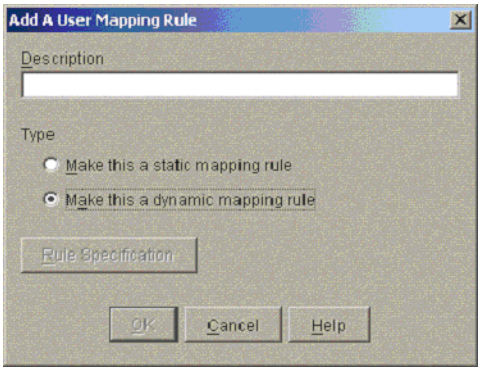
Figure 45 shows the User Mapping page:

Figure 45 samlTrustedAffiliate: User Mapping Page



Click the plus sign (+) on the right side to launch the Add A User Mapping Rule dialog box, as shown in Figure 46:

Figure 46 Add a User Mapping Rule Dialog Box

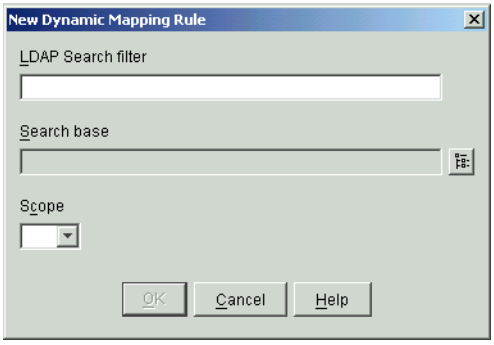


This dialog box allows the administrator to give the new user mapping rule a description and decide whether dynamic or static user mapping rule is created.

Dynamic User Mapping Rules

To create a dynamic user mapping rule, provide a description, select Make This a Dynamic Mapping Rule, then click Rule Specification. The dynamic user mapping dialog box launches, as shown in Figure 47:

Figure 47 New Dynamic Mapping Rule Dialog



The LDAP Search filter contains the LDAP search string that performs the search. The SAML Attributes that you want to search against are defined by using the SAML (attribute name) key word. For example, if you want to perform a search against the LDAP mail attribute and you have an incoming SAML Attribute named Email, the search string is:

```
(mail=SAML(Email))
```

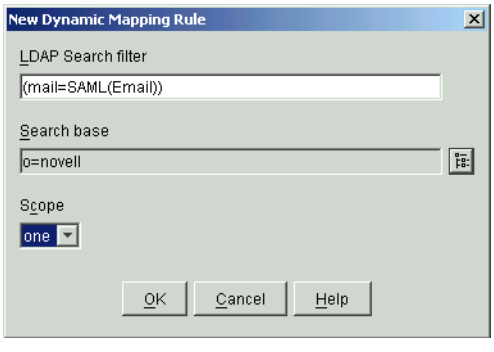
When this search is evaluated for an incoming user, the SAML(Email) value is replaced with the SAML Email attribute. For example, if the SAML assertion contains an Email attribute of joe@excite.com, the resulting search string is:

```
(mail=joe@excite.com)
```

The search base defines the organization or organizational unit where the search begins. The scope defines whether the sub-tree of the search base is included in the search. If the value one is used, only the container specified in the Search base is searched. If sub is used, the Search base and all its subcontainers are searched for a match.

The following figure shows the complete definition of the Email search rule:

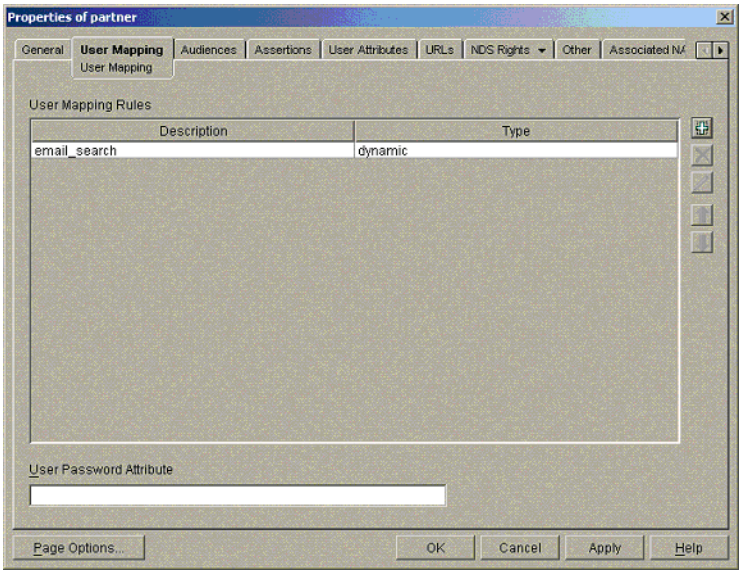
Figure 48 New Dynamic Mapping Rule: Email Search Rule



This rule performs a search matching users' mail attribute with the incoming SAML assertion's email attribute. The search begins in the o=novell container, but sub-trees are not searched.

After you click OK, the User Mapping page now has a rule entry, as shown in Figure 49:

Figure 49 User Mapping Page: Rule Entry



Static User Mapping Rules

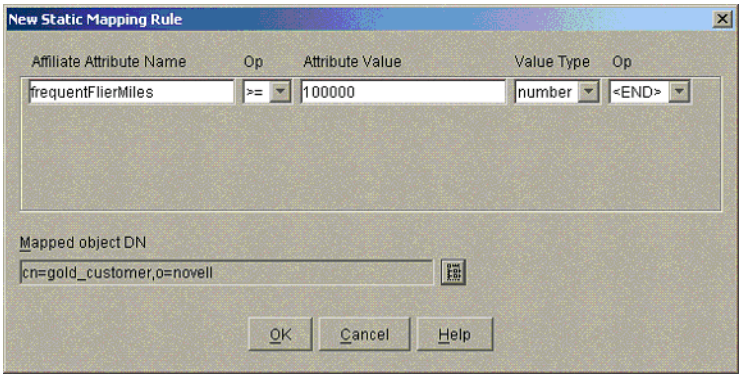
Static user mapping rules take attributes in the SAML assertion and compare them against statically defined values. If the comparison evaluates to TRUE, then a statically defined user reference is used as the mapping. Static user mapping rules are ideal to map many users into a smaller set of groups or roles. For example, an airline may have four different customer categories:

- Gold: 100,000 or more frequent flyer miles
- Silver: 50,000 or more frequent flyer miles
- Bronze: 10,000 or more frequent flyer miles
- Lead: Less than 10,000 frequent flyer miles

Consider that two airlines are sharing customers with each other. When users are sent to the airline's portal, the airline requests that the number of frequent flier miles are sent as a SAML Attribute called frequentFlierMiles. This way, the airline portal can present the user with different offers and content based upon his or her customer category. This type of mapping is ideal for a set of static user mapping rules.

To create a new mapping rule, select the plus sign (+). The New Mapping Rule dialog box is displayed. Select Static Mapping Rule > Rule Specification. The static user mapping dialog box is shown. Figure 50 contains the gold user mapping rule, as shown:

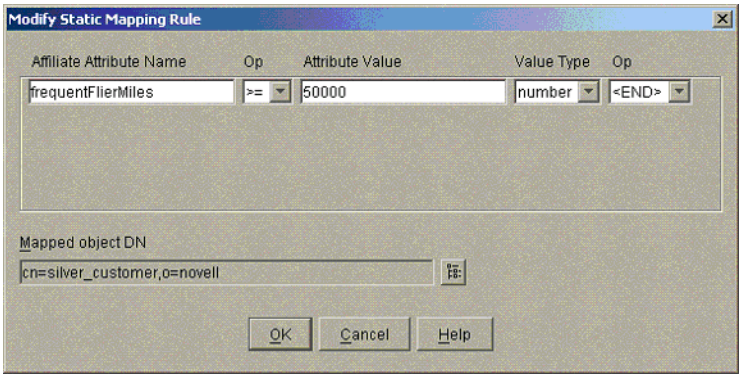
Figure 50 New Static Mapping Rule: User Mapping Rule



This rule states that if the frequentFlierMiles SAML attribute is greater than or equal to 100,000, the user is mapped to cn=gold_customer,o=novell user object.

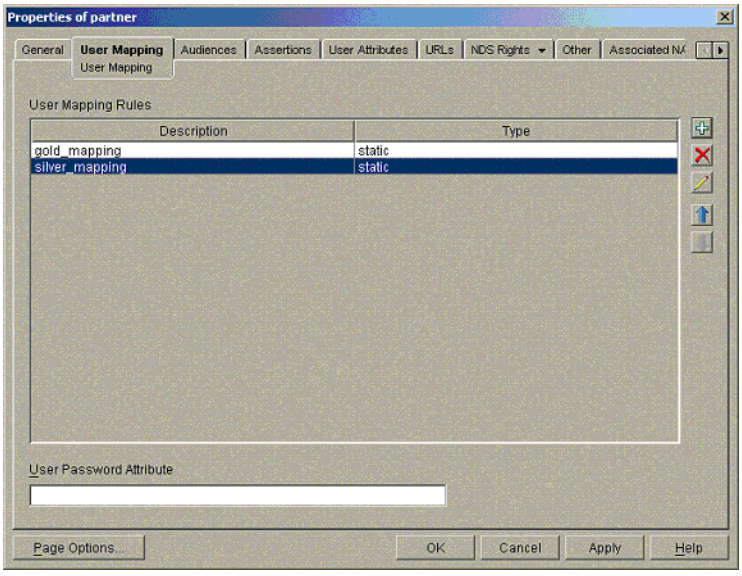
Next, define the mapping rule for silver users, as shown in Figure 51:

Figure 51 Modify Static Mapping Rule



The upper bound on this rule has not been set, meaning that if a gold user were to be evaluated with this rule, he or she would be mapped to cn=silver_customer,o=novell rather than the correct gold user. This shows the importance of rule ordering. If you place the gold user before silver in the list, you are assured that gold users are always be successfully mapped by the gold rule *before* the silver rule is ever evaluated. Figure 52 shows the user mapping table:

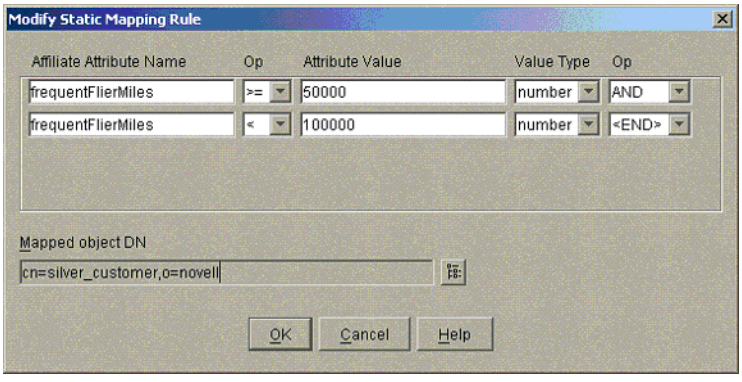
Figure 52 Rule Ordering



Take note of the blue arrow icons on the right ride of the panel. Use these arrows to change the order of the rules in the list. The upward-pointing arrow moves the rule higher up in the list so that it will be evaluated sooner, and the downward-pointing arrow moves the rule down in the list so that it is evaluated later.

If you want to be certain that the silver user mapping rule didn't accidentally map gold users to silver, you can add an additional constraint (upperbound) onto the silver mapping rule as shown in Figure 53:

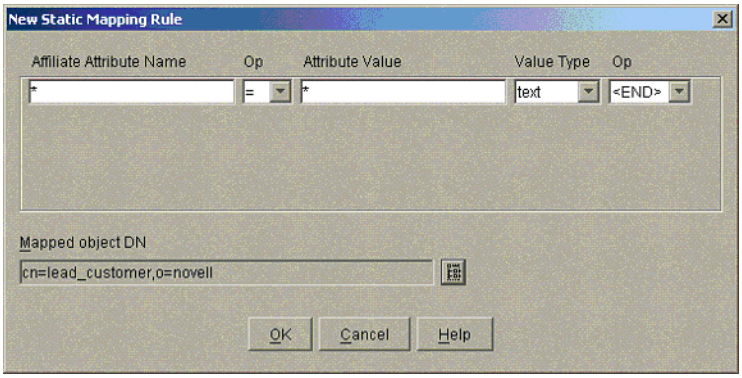
Figure 53 Mapping Rule: Adding an Additional Constraint



This rule ensures that the number of frequent flier miles is less than the gold threshold so that no gold users will be erroneously mapped to silver.

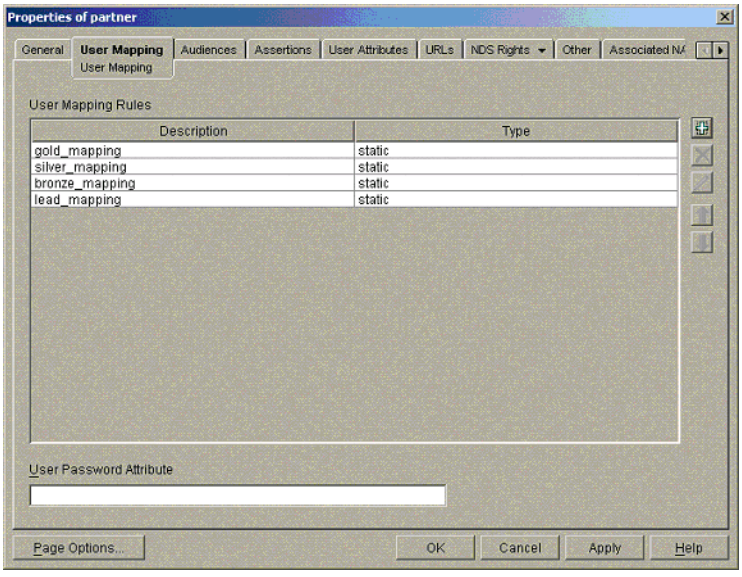
A similar rule can be created for bronze. You can create a special wildcard rule for the catch-all lead user. For lead, it doesn’t really matter how may frequentFlierMiles the user has. Because he or she already failed the mapping test for gold, silver, and bronze, you know that he or she must be considered as lead. An asterisk (*) character placed in the attribute or value name causes the rule to always evaluate to TRUE, as shown in the lead mapping rule in Figure 54:

Figure 54 Mapping Rule: Wildcard Rule



The resulting user mapping list is shown in Figure 55:

Figure 55 User Mapping List



According to the ordering of the mapping rules used in this example, the rules will always be evaluated in the following order:

1. gold_mapping
2. silver_mapping
3. bronze_mapping
4. lead_mapping

Mixing User Mapping Rule Types

Static and dynamic user mapping rules can be used in the same user mapping rules list. Generally, it is a good idea to include a wildcard user mapping after a dynamic user mapping rule to prevent cases where no results are found. Dynamic user mapping rules follow the same ordering rules as the static rules.

Default User Mapping Rule

If the user mapping rules table is left blank, the Default user mapping defined in the samlSiteConfig object is always performed. If the Default user mapping is blank, no one from the specified Trusted Affiliate can access your site because the user mapping process always fails.

User Password Attribute

Some applications fronted by iChain still require a user password in order to function properly. This value allows the administrator to specify an incoming SAML attribute value to use as the user's password when the session is created on iChain. For instance, if the Trusted Affiliate sends the user's password as a SAML Attribute named UserPwd, by placing UserPwd in this field, the resulting iChain session creates using the UserPwd value.

NOTE: This feature is included to support legacy applications that require a user password for authentication. You should avoid using it if possible.

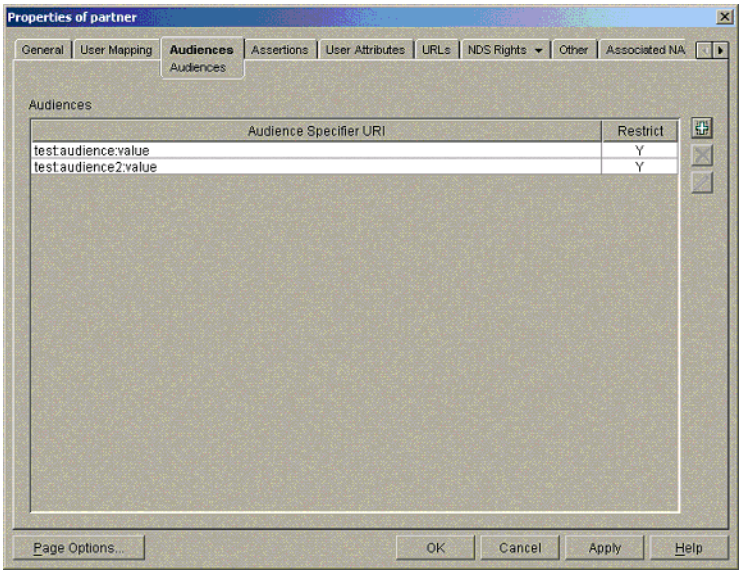
Audience

You can address a SAML assertion to any specific audience. This audience value specifies which audiences are accepted for incoming assertions from the affiliate. If an audience is marked as "restrict," all assertions outbound to this affiliate include the audience value in the audience restriction.

The value of an audience is a URI reference that identifies an intended audience. The URI reference could even identify a document that describes the terms and conditions of audience membership. In the Liberty specification, which uses SAML assertions, the audience that is used is the provider ID. When you first set up a SAML partnership, adding audience restrictions conditions is probably unnecessary and could add complexity.

Figure 56 shows an example where SAML assertions are accepted with the test:audience:value or test:audience2:value, and a SAML Audience restriction condition is added:

Figure 56 Audience Page

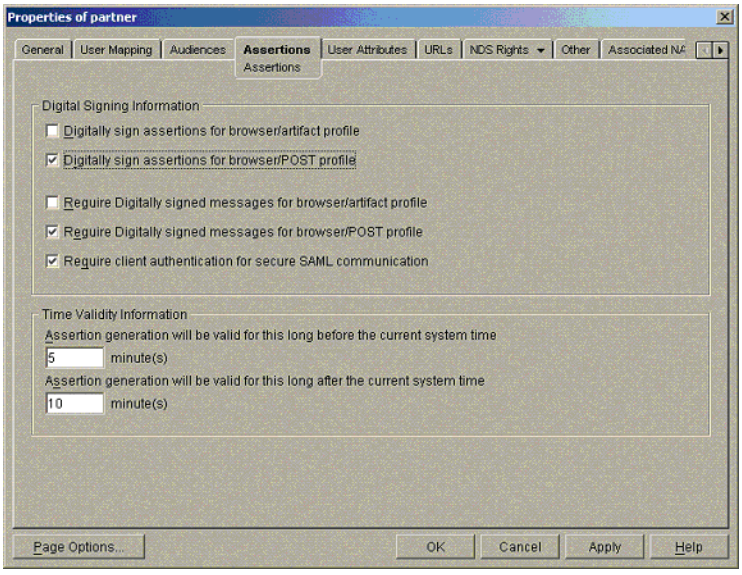


For more information about how SAML Audiences restriction conditions are used in the SAML protocol, please refer to the SAML specification documents.

Assertions Tab

The Assertions page contains settings relating to security constraints placed on assertions that are accepted from the Trusted Affiliate site, and how to generate SAML assertions for the Trusted Affiliate. Figure 57 shows the Assertions page with the default settings applied:

Figure 57 Assertions Tab: Default Settings



The first two settings apply to SAML assertions that you generate for this Trusted Affiliate for outgoing users:

Digitally Sign Assertions for Browser/Artifact Profile: Indicates whether a digital signature should be generated when creating SAML assertions intended for this Trusted Affiliate using the browser/Artifact profile. The SAML specification does not require that browser/Artifact SAML data be signed, so the default value is set to False.

Digitally Sign Assertions for Browser/POST Profile: Indicates whether a digital signature should be generated when creating SAML assertions intended for this Trusted Affiliate. The SAML specification requires that browser/POST SAML data be signed, so the default value is set to True.

The next three settings apply to SAML assertions that you accept from this Trusted Affiliate for incoming users:

Require Digitally Signed Messages for Browser/Artifact Profile: Indicates whether you accept SAML assertions from this Trusted Affiliate using the browser/Artifact profile if they are not signed. Because the SAML specification does not require that browser/Artifact SAML data be signed, the default value is set to False. If the value is True and a browser/Artifact SAML assertion is received without a digital signature, it is rejected and the incoming user cannot access your site.

Require Digitally Signed Messages for Browser/POST Profile: Indicates whether you accept SAML assertions from this Trusted Affiliate using the browser/POST profile if they are not signed. Because the SAML specification requires that browser/POST SAML data be signed, the default value is set to True. If the value is True and a browser/POST SAML assertion is received without a digital signature, it is rejected and the incoming user cannot access your site.

This setting applies to both the incoming and outgoing halves of the SAML relationship:

Require Client Authentication for Secure SAML Communication: The SAML back-channel can be run over HTTPS with or without client authentication. The SAML Extension server has two separate URLs to handle SSL and SSL-M. These are:

- **https://<host>/cmd/ext/samlext/saml/resp:** This is for non-client-authenticated communication.
- **https://<host>/cmd/mutExt/samlext/saml/resp:** This is for client-authenticated communication.

Because the SAML specification requires client authentication, the default value for this setting is True. If communication requests are received over the non-mutually authenticated channel (/cmd/ext) and this setting is True, the communication request is rejected.

The final two settings relate to SAML assertion generation, and relate to the outbound user half of the relationship:

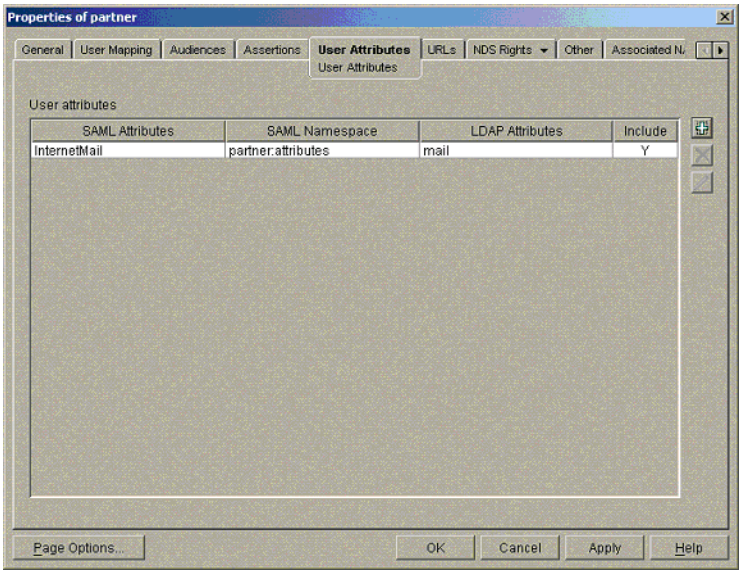
Assertion Generation Will Be Valid for this Long Before the Current System Time: Allows the default value defined in the samlSiteConfig object to be overridden for this Trusted Affiliate. For details on what the setting does, see “Assertions” on page 46.

Assertion Generation Will Be Valid for this Long After the Current System Time: allows the default value defined in the samlSiteConfig object to be overridden for this Trusted Affiliate. For details on what the setting does, see “Assertions” on page 46.

User Attributes

The user attributes page applies only the out-bound aspect of the SAML trust relationship. On this page, the SAML administrator defines what attributes are made available to SAML Trusted Affiliate sites, as well as which attributes are included in SAML single sign-on Assertions. Figure 58 shows the User Attributes page:

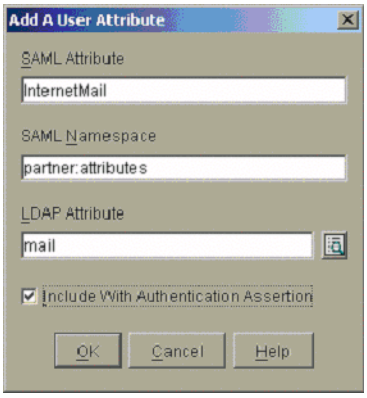
Figure 58 User Attributes Page



Typically, the names and namespaces for these attributes are requested by the Trusted Affiliate site. The Trusted Affiliate requests, for instance, that the user's e-mail address be made available and that it be named “InternetMail” with namespace “partner:attributes” in the SAML assertion. Click

the plus sign (+) to create a new attribute. The Add a User Attribute dialog box launches, as shown in Figure 59:

Figure 59 Add A User Attribute Dialog Box



SAML Attribute: The SAML attribute name value in the generated SAML assertion.

SAML Namespace: The SAML attribute namespace value in the generated SAML assertion.

LDAP Attribute: The LDAP attribute value that is to be queried and used as the SAML attribute value.

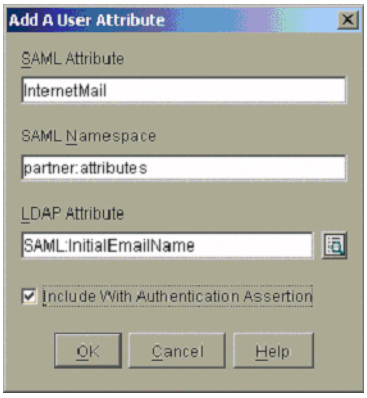
Include With Authentication Assertion: Indicates whether this attribute is to be sent with SAML authentication assertions generated for this Trusted Affiliate. Generally, unless the partner site requests the SAML attributes in a separate SAML attribute query, this caption should be selected.

The SAML extension server can also respond to SAML attribute queries. The SAML Extension server provides attributes included only in this User Attributes list.

Advanced Usage: Overriding LDAP Attributes

In some cases, the user for which you are generating the SAML assertion didn't directly access your site, but rather, single signed-on using SAML. The initial SAML assertion used to authenticate to the site could have included some attribute values about the user. If the user then wants to access another site, you might want to use the attributes from the initial SAML assertion for the user rather than attributes read from the directory. The SAML extension server for Novell iChain stores the initial SAML attribute values so they are available for outbound assertion generation. This is done by creating a User Attribute as shown in Figure 60:

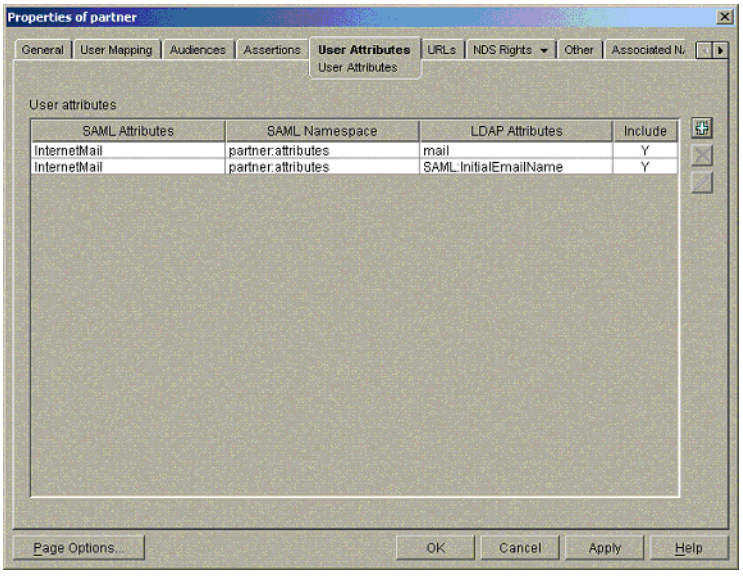
Figure 60 Overriding LDAP Attributes



In this example, you would override the LDAP mail attribute with the initial SAML attribute named InitialEmailName. The system recognizes that this is a cached SAML attribute because of the SAML: prefix on the LDAP Attribute name. In searching for the cached SAML attribute, the SAML: prefix is removed, so in this example, the initial SAML attribute InitialEmailName is used as the outgoing InternetMail SAML assertion Attribute.

The resulting User Attributes table is shown in Figure 61:

Figure 61 User Attributes Table: Overriding the LDAP Mail Attribute

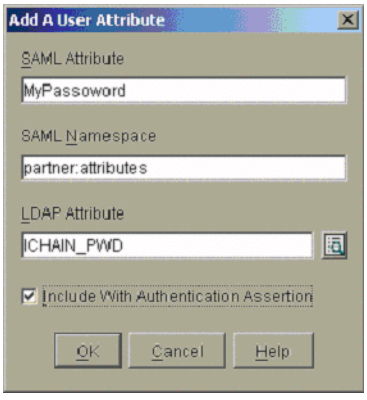


Password Attribute

As mentioned in “User Mapping” on page 51, some applications still require a user password in order to function properly. When performing SAML single sign-on, the destination site receives no user password, so some applications might not function properly. To get around this, the ability to pass the user password as a SAML attribute has been added. This attribute should not be used unless absolutely necessary, because the user password is exposed to the partner site. Furthermore, the password attribute should only be sent when using the browser/Artifact profile, since the

browser/Artifact passes the SAML assertion directly between the sites using SSL. The password attribute is specified by placing ICHAIN_PWD as the LDAP Attribute name, as shown in Figure 62:

Figure 62 Specifying the Password Attribute



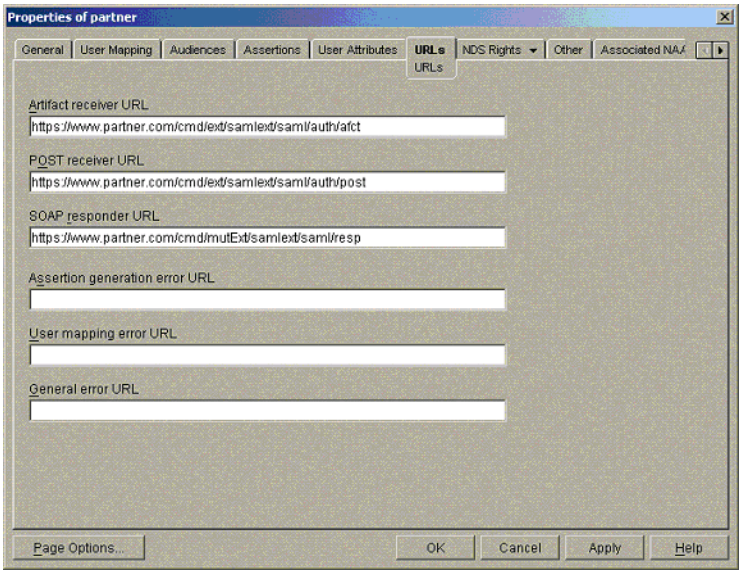
The ICHAIN_PWD can also be used by OLAC at the receiving site to pass the password to the back-end Web server as a query string or header parameter.

In order for a SAML Extension for iChain partner site to use this attribute, the User Mapping page would need to have its User Password Attribute setting equal My Password.

URLs

The URLs page is the most important page in the Trusted Affiliate configuration. This page is where you enter the information that tells the SAML extension server for Novell iChain service where to send or redirect users when a SAML event or error occurs.

Figure 63 samlTrustedAffiliate: URLs Tab



The sample data in the above graphic is valid for a Trusted Affiliate site running the SAML Extension server for iChain with host name =www.partner.com.

Artifact Receiver URL: The user is redirected to this URL when a Browser/Artifact authentication is requested between this site and the partner site.

POST Receiver URL: The user is sent to this URL when a Browser/POST authentication is requested between this site and the partner site.

SOAP Responder URL: The SOAP endpoint that this site sends SAML artifact requests to when authenticating users from this partner site using the Browser/Artifact profile. In Figure 63, the mutually authenticated communications channel is being used (notice the /cmd/mutExt URL extension). To use the non-client authentication channel, replace /cmd/mutExt with /cmd/ext.

Assertion Generation URL: If an error occurs during SAML assertion generation, the user is sent to this URL. If there is no value specified, the General error URL or samlSiteConfig Default error URL is used.

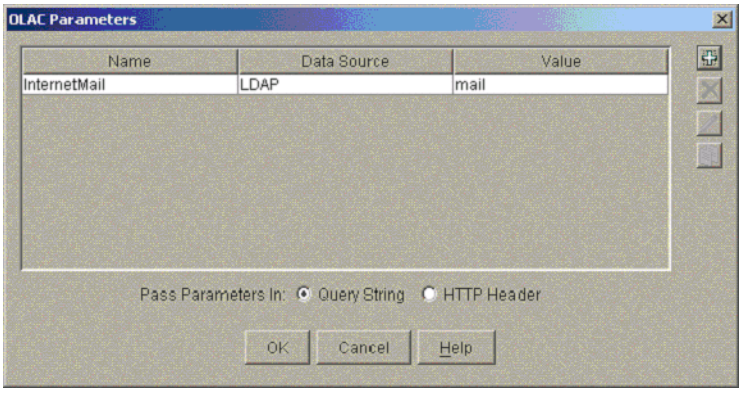
User Mapping Error URL: If an error occurs during the SAML user mapping processes, the user is sent to this URL. If there is no value specified, the General error URL or samlSiteConfig Default error URL is used.

General Error URL: If an error occurs during assertion validation for SAML data received from this affiliate, or if other error URLs are not set, the user is sent to this URL. If this value is not set, the samlSiteConfig Default error URL is used.

Accessing SAML Attributes in OLAC

As previously mentioned, SAML assertions can contain user attribute information. When SAML attributes are found in incoming SAML assertions, the attribute data is cached by iChain for later use in object level access control (OLAC). As with LDAP attributes, SAML attributes can be sent to back-end Web servers as OLAC parameters. This is done by specifying SAML as the OLAC DataSource and the SAML Attribute name as the OLAC Value. For example, consider that you have a protected resource where you are sending the users email address as an OLAC parameter, as shown in Figure 64:

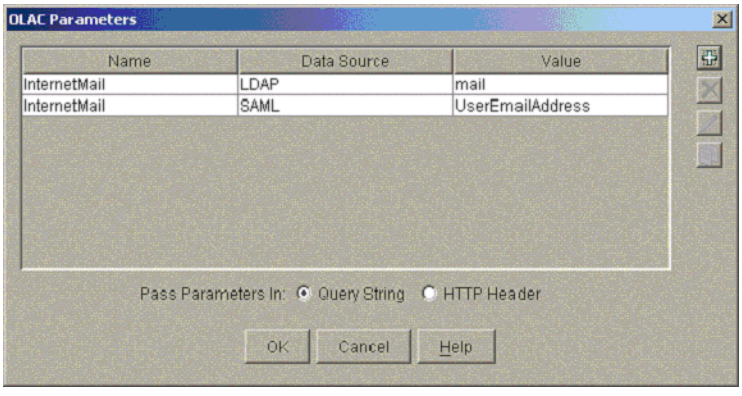
Figure 64 Protected Resource Using User E-mail as OLAC Parameter



In this example, the OLAC parameter reads the mail LDAP attribute and sends it as InternetMail to the back-end Web server.

In this example, you want the OLAC parameter to be available for a user who logs directly into the protected resource using the directory. However, a user who accesses the site via a SAML assertion has a SAML attribute that has his or her e-mail address, so when he or she accesses the protected resource with SAML, you want the SAML value to be used instead. This is done by creating another OLAC parameter definition with the same name as the first, InternetMail. The DataSource is now SAML rather than LDAP, and the value is the name of the SAML attribute in the assertion. If you were receiving the SAML attribute named UserEmailAddress, the OLAC parameter would be as shown in Figure 65:

Figure 65 OLAC Parameter When SAML Attribute Is UserEmailAddress



When the user accesses the site using SAML, the value of the SAML UserEmailAddress attribute is used in place of the LDAP mail attribute. If no SAML UserEmailAddress attribute is available, the LDAP mail address is used.

SAML-Specific Attributes

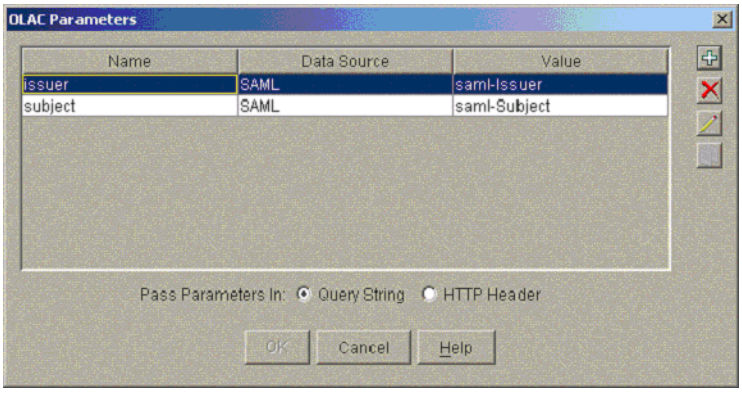
The following SAML-specific attributes are available:

saml-Issuer: Contains the SAML Issuer ID of the assertion used to authenticate the user.

saml-Subject: Contains the original SAML Subject name contained in the SAML assertion.

These attributes can be accessed in OLAC as shown in Figure 66:

Figure 66 Accessing SAML-Specific Attributes in OLAC



4 Configuring the SAML Extension Server

This section contains the following topics:

Generating the Configuration File

When the SAML extension server is installed, a configuration file is generated and placed in the samlext/conf directory on the local drive of the server. This file is named samlextConfig.xml. The SAML extension server reads most of its configuration information from eDirectory™. However, some basic information must be made available to the SAML extension server before a connection to the directory is established. Also, some sensitive information cannot be safely transferred or stored between eDirectory and the SAML Extension server. For this purpose, samlextConfig.xml contains several key pieces of information:

- Addresses of the eDirectory servers containing user and configuration information
  - Proxy user and password
  - iChainServiceObject associated with the system
- Public Key Certificate of eDirectory servers if LDAP over SSL is to be used
- Key store containing the XML-Signature key pair
- Key store containing the SSL - Mutual client key pair
- Debug page enable flag

The following shows an example of what a typical samlextConfig.xml file looks like:

```
<authority name="SAMLExtDirectoryAuthority">
  <class>

<name>com.novell.wss.authority.saml.SAMLExtDirectoryAuthority</name>
  </class>
  <data>

    <servers refresh="1">
      <url>ldap://137.65.159.66:389</url>
    </servers>

    <proxyUser>
      <dn>cn=admin,o=novell</dn>
      <password>novell</password>
      </proxyUser>

    <initialCapacity>10</initialCapacity>
    <maxCapacity>30</maxCapacity>
```

```
    <isoDn>cn=iso_ichainsite,o=novell</isoDn>

    <keypairs>
      <keypair usage="ssl" type="pkcs">
         <password>dude</password>
         <file>ssl.pfx</file>
      </keypair>

      <keypair usage="signing" type="jks">
         <password>dude</password>
         <file>sig.pfx</file>
      </keypair>

    </keypairs>

    <enableDebugPages>false</enableDebugPages>

  </data>

</authority>
```

Configuration Elements

- authority, class, name (required): These elements and associated attributes should not be modified.
- data (required): Contains the basic configuration information for the SAML Extension server.
- refresh: The refresh attribute determines how often (in minutes) the health check thread will run on the LDAP server connection pools. If a server goes down, the health check monitor will deactivate it. If a server that was previously down becomes available, the health check will make those connections available again. The default value of 4 minutes will be used if no refresh attribute value is provided.
- servers (required): Allows the administrator to specify one or more eDirectory servers from which to read configuration and user attribute information. If more than one server is specified a “round-robin” approach is used to accomplish a type of load balancing between the servers. Also, a health check is performed on each of the servers at a specified time interval, and servers that are down are removed from the pool.
  - refresh: Determines how often (in minutes) the health check thread runs on the LDAP server connection pools. If a server goes down, the health check monitor deactivates it. If a server that was previously down becomes available, the health check makes those connections available again. The default value of 4 minutes is used if no refresh attribute value is provided.
  - url (required): Specifies a single LDAP URL.
- proxyUser,dn, password (required): These elements specify the user and password to use when making connections to eDirectory. The specified user must have read privileges to the iChainServiceObject.
- initialCapacity (required): The number of connections to the specified directory to create and make available to the system at startup time.
- maxCapacity (required): The maximum number of connections to create to a given directory. When requests are made to the system to obtain a LDAP connection and all are in use, new connections are created and added to the available pool up to the maxCapacity value.

- isoDn (required): The distinguished name of the iChainServiceObject associated with the iChain® site. This setting allows the system to locate its configuration objects in the directory.
- keypairs (optional): Specifies public-private key pair objects to use for creating digital signatures on SAML data and creating outbound SSL connections. Applicable attributes are:
  - usage (required): Two values are appropriate; SSL indicates that the keypair should be used to create outbound SSL connections, and SIGNING indicates that the key pair should be used to create digital signatures on SAML data.
  - type (optional): The type of keystore to open. Valid values are PKCS or PKCS12 for PKCS#12 keystores or JKS for Java Key Store keystores.
  - alias (optional): JKS keystores can contain more than one public-private key pair. The alias attribute allows the administrator to specify which key pair to use. (This value is only used for the signature). If this value is absent, the first valid keypair found in the store is used.
  - keyPassword (optional): JKS keystores (JKS) can have a separate “key” password for each key in the keystore. If this value is absent, the keystore password is used for this value.
- file (required element): The filename of the keystore. The filename can be named relative to the location of the samlextConfig.xml file.
- password (required element): The password used to load the keystore.
- enableDebug (optional, default=false): Allows the administrator to specify whether the SAML Extension debug pages are available. It is generally a good idea to enable the pages when first deploying and debugging the system. However, as soon as the system is put into production or accessed by non-administrators, these pages should be disabled. There are three debug pages available:
  - /samlext/status: Displays the status of the system including the LDAP connection pools and currently loaded SAML configuration. There is a possibility of a security issue because some sensitive information can be displayed.
  - samlext/test: Displays the attributes of the currently connected user. It shows all of the incoming headers and parameters included in the HTTP request as sent by iChain.
  - samlext/saml/test: Displays the SAML configuration as loaded by the system.

Applying LDAP SSL Settings

In order to communicate with the LDAP server over SSL, you must apply additional settings. First, the LDAP SSL public key certificate for each directory you want to connect to via SSL must be copied onto the local drive of the SAML extension server. The samlextConfig.xml file must point to these certificates so that they can be added to the Java SSL TrustStore, enabling the system to make SSL connections to those servers. Second, a single directory server entry must be specified as the configuration server that the system can read initial configuration from before upgrading the connections to SSL. The following example shows the differences between the non-SSL and SSL configuration files:

```
<authority name="SAMLExtDirectoryAuthority">
  . . .

    <configServers>
      <url>ldap://137.65.159.66:389</url>
    </configServers>
```

```
    <servers>
      <url secure="true">ldap://137.65.159.66:636</url>
      <url secure="true">ldap://137.65.159.83:636</url>
    </servers>

    <ldapCertificates>
      <ldapCertificate>
        <file>dir_66.der</file>
      </ldapCertificate>
      <ldapCertificate>
       <file>dir_83.der</file>
      </ldapCertificate>
    </ldapCertificates>
 . . .

  </data>
</authority>
```

SSL-Enabled Configuration Elements

configServers (required): If secure LDAP connections are to be used, the configServers element must contain at least one LDAP server. The system uses this LDAP server to read its basic configuration information before transitioning to the SSL connections. After startup, the configServer connections are all closed.

servers (required): The servers listed here can have the secure attribute set to True. This causes the connections to be made using SSL.

ldapCertificates, ldapCertificate (required): The specified certificates are added to the Java SSL truststore so that SSL connections can be made to the specified LDAP servers

5 SAML Security Considerations

The SAML specifications document contains an excellent discussion of the security concerns and considerations relating to the SAML protocol and profiles. The document is available on the OASIS Web site (http://www.oasis-open.org).

This chapter discusses how to configure the SAML system to generate and validate XML digital signatures. Digital signatures allow the receiving site to validate that the source of the SAML assertion really was the referring site, and that the contents of the SAML assertion were not changed in transit. Also discussed is how to configure SAML to use SSL with client authentication on the SAML SOAP back-channel. The inclusion of SSL with client authentication means that both the server receiving the incoming request and the client making the request must present certificates validating their identity. The advantage for SAML in this scenario is that the source of a SAML assertion can be validated by the SSL layer, and does not necessarily need to contain an XML digital signature.

This chapter is divided into the following sections:

Generating SAML Digital Signatures

The SAML specification allows the inclusion of XML digital signatures on generated SAML Responses and SAML assertions. The SAML browser/POST (where SAML assertions travel through the user's browser between sites) requires that SAML data contain an XML signature. The SAML browser/Artifact (where SAML assertions are sent directly server to server via the SAML SOAP back-channel) do not necessarily need to be signed.

Security settings relating to digital signatures are made on a per-Trusted Affiliate basis. This means that for each separate Trusted Affiliate relationship you can decide:

- If digital signatures are generated on outbound SAML assertions intended for the Trusted Affiliate
- If digital signatures are required on incoming SAML assertions issued by the Trusted Affiliate
- What public key certificates to use to validate digitally signed incoming data from the Trusted Affiliate

These settings determine the level of security that is required for the relationship, and they must be negotiated between your site and each Trusted Affiliate site. The settings that control this are found on the Assertions page for the samlTrustedAffiliate configuration objects in the directory.

Use the following rules to configure the SAML system to generate digital signatures:

- A signing key pair (SKP) must be generated using Novell® Certificate Server™ or another PKI management tool.
- The SKP can exist in eDirectory™, in which case you can use the Novell Certificate Server utilities to manage it.

  If the SKP exists in eDirectory, you can point to it in the samlSiteConfig object.
- The SKP is made available in PKCS#12 or Java Key Store format and stored on the SAML extension server's local disk.

  Novell Certificate Server exports the key pair in PKCS#12 format.
- The Appropriate settings must be made on the Trusted Affiliate object whose SAML data you want to sign.
- The public key certificate associated with the SKP must be exported and sent to the Trusted Affiliate that will be validating the signatures you generate.

  Be sure that the public key certificate is exported, not the “parent” Trusted Root. You must have the key that directly corresponds to the signing key; this is the public key certificate.

Creating a Signing Key Pair

There are three general ways to generate or import a key pair into an eDirectory system. You can:

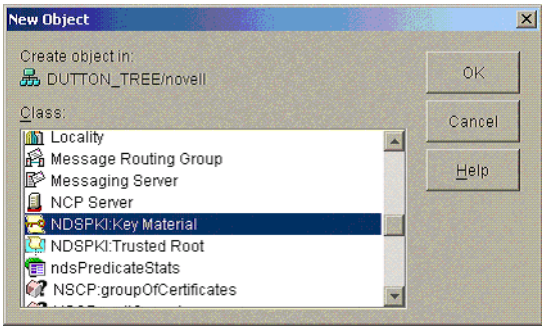
- Use Novell Certificate Server to generate the key pair and certificate, signed by your Tree CA.
- Use Novell Certificate Server to generate the key pair and Certificate Signing Request to be signed by an external CA.
- Use Novell Certificate Server to import an external key pair stored in PKCS#12 format.

The following instructions show how to generate a key pair using Novell Certificate Server and the Novell Certificate Server snap-ins. For more detailed information on key and certificate management, *Novell Certificate Server 2.7.x Administration Guide* (http://www.novell.com/documentation/crt27/index.html).

1. Create a NDSPKI:Key Material object.

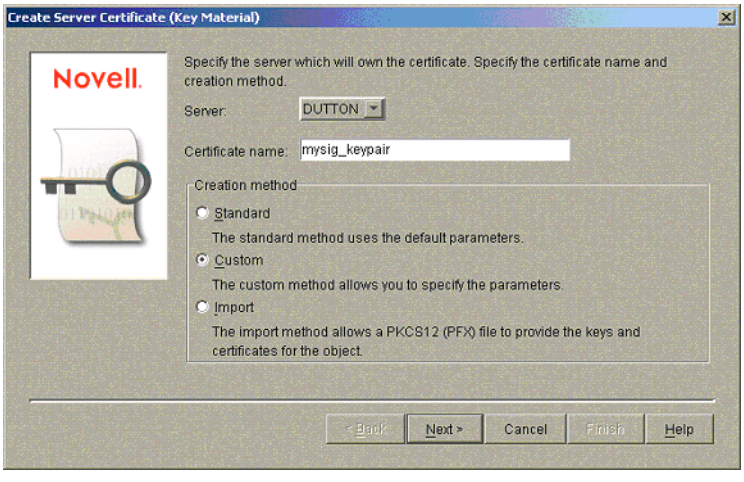
   This is what Novell Certificate Server calls a public - private key pair object. Novell Certificate Server stores and associates key pair objects with Servers, so you must create the Key Material object in the same container as the server where you want to host the Key. For SAML extension purposes, the server you choose is immaterial. Figure 67 shows the Key Material selection:

Figure 67 NDSPKI:Key Material Object



2 Click NDSPKI:Key Material Object, then click OK. A wizard guides you through the certificate generation process.

Figure 68 Certificate Generation Wizard



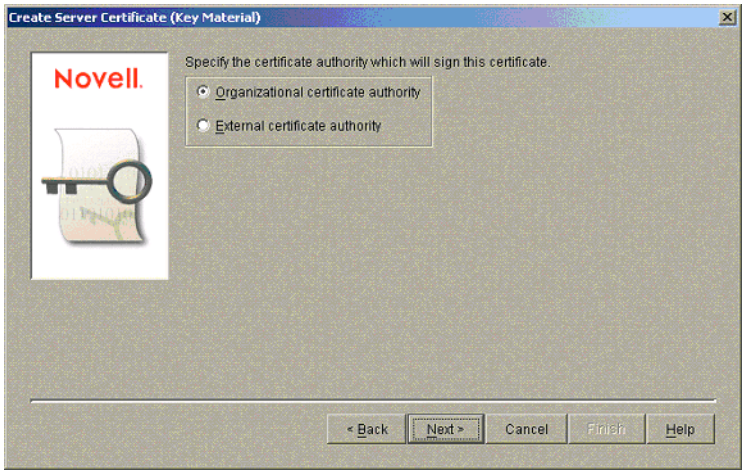
3 Select the server you want to create the private key on.

The Creation method selector determines what type of operation you are going to perform. If you already have a certificate you want to use, select the Import option. Otherwise, select Custom. If you choose the Import option, you are prompted to enter the filename and password associated with the PKCS#12 file you want to import. See the *Novell Certificate Server 2.7.x Administration Guide* (http://www.novell.com/documentation/crt27/index.html) for details on importing external certificates.

4 Click Next.

5 Decide whether to create a public key certificate signed by your Tree's Certificate Authority (CA) or an external CA.

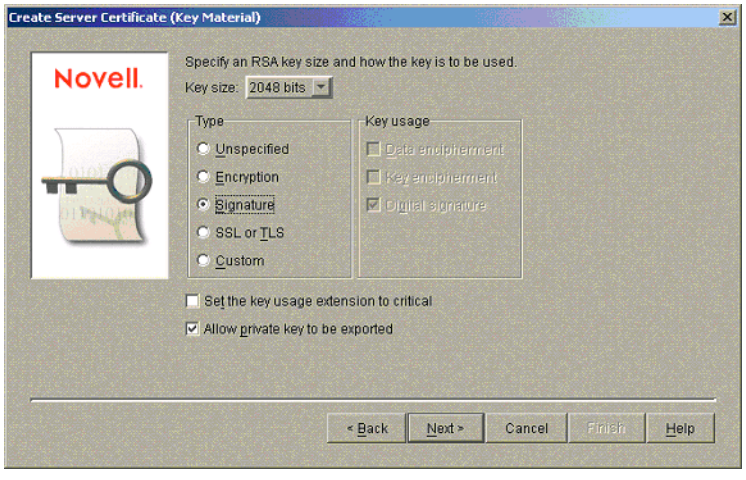
Figure 69 Create Server Certificate: Key Material



Generally, if you are creating a certificate for testing, you can use your built-in Organizational Certificate Authority. For production, you will probably want a certificate signed by a well-known CA such as Verisign* or Entrust*. Select External Certificate Authority if this is what you want to do.

6 Click Next.

Figure 70 Key Pair Properties



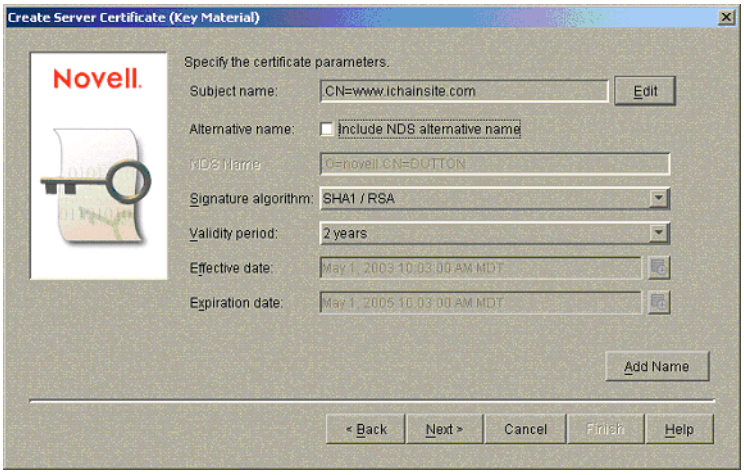
7 Define the key pair properties. Because this certificate is to be used to sign SAML data, verify that the Signature Type value is selected. Generally, a key size of 2048 bits is sufficient.

8 Click Next.

9 Define additional key and certificate properties.

Make sure you create a subject name that allows your partner sites to identify the certificate as yours. A general rule is to make sure the subject name is the same as your Site ID. For example, if you are generating a certificate for the ichainsite sample site, the Subject name is .CN=www.ichainsite.com. Subject names in the Novell Certificate Server must begin with a period (.), as shown in Figure 71:

Figure 71 Certificate Parameters



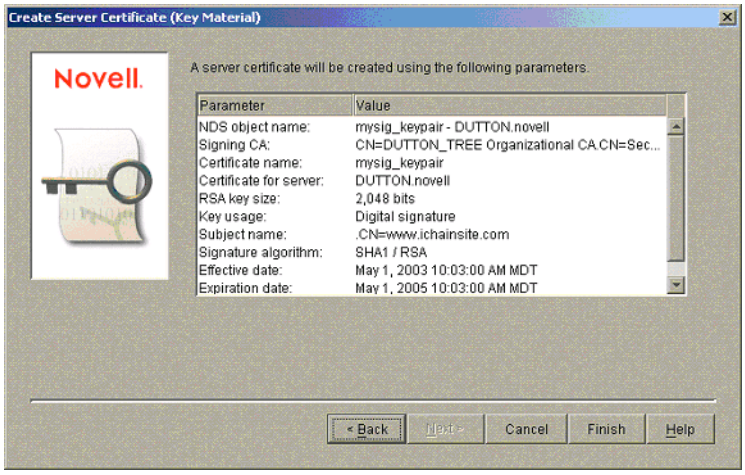
10 Click Finish to complete the operation.

If you are signing the certificate with your Tree CA (Organizational), you are prompted to select which certificate will sign your certificate. Either selection will work for testing purposes. For information about the differences between the two, refer to the *Novell Certificate Server 2.7.x Administration Guide* (http://www.novell.com/documentation/crt27/index.html).

Figure 72 Specify the Trusted Root

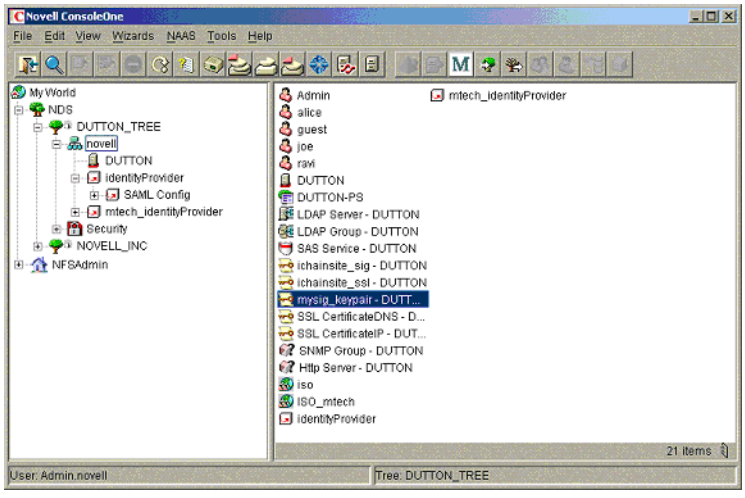
11 You are presented with a summary page outlining all of the selections that you made in the wizard. If you are satisfied with all of your selections, click Finish. The key pair is generated.

Figure 73 Summary Page



You should have a new NDSPKI: Key Material Object in the directory. The name of this new object is the name you specified in the Wizard, followed by the hosting server name. Figure 74 shows the key pair generated. In this example, the certificate name is mysig_keypair - DUTTON.

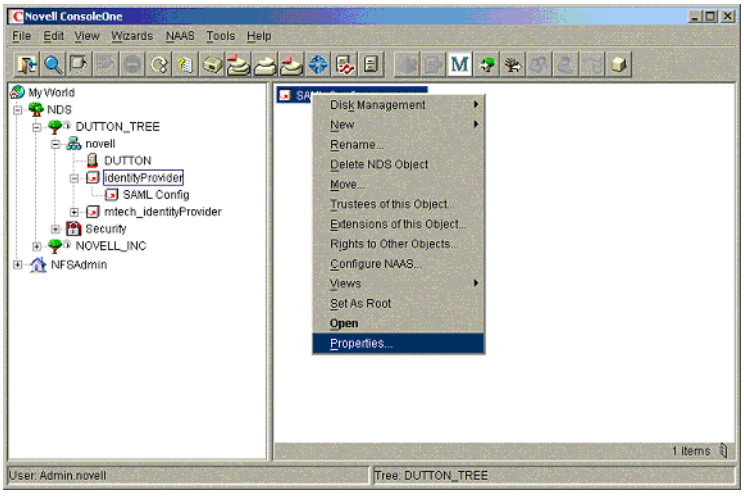
Figure 74 NDSPKI: Key Material Object Shown in Directory



12 (Optional) Associate your SAML Configuration object with this key pair object.

This step is not mandatory, but it is recommended because it keeps the association between the key pair you generated and the SAML configuration. Create this link by selecting the SAML Configuration object > Properties. See Figure 75:

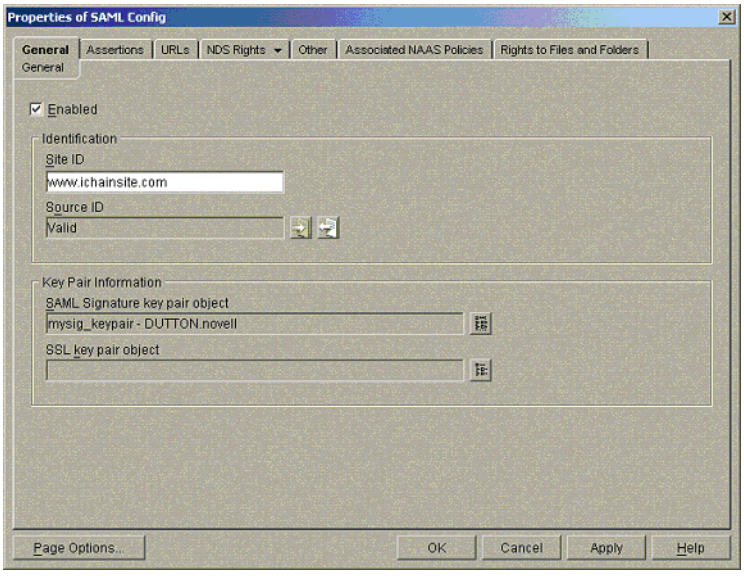
Figure 75 SAML Configuration Object: Properties



13 Click the General tab.

In the key pair Information section of the page, you can specify the key pair object that you are using to sign SAML data. Figure 76 shows this selection:

Figure 76 General



Now when you are working with the configuration of the SAML system, you have a link back to the key pair you are using to sign SAML data.

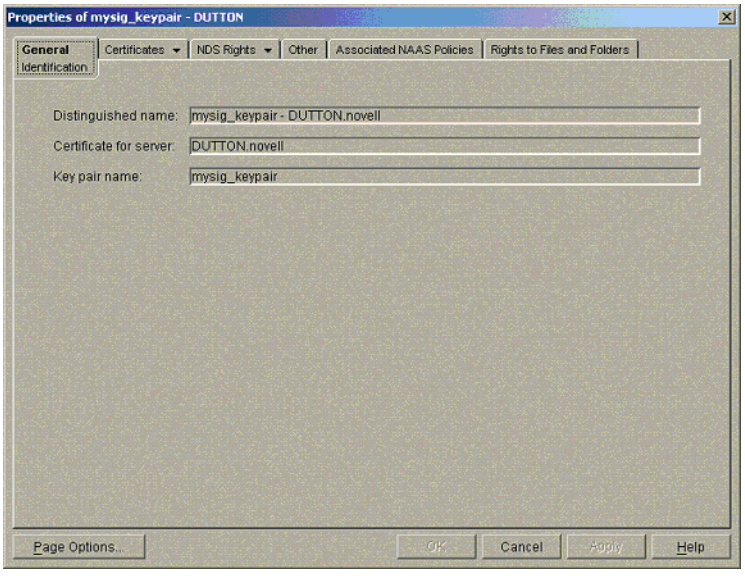
Exporting a Signing Key Pair

After you have created the signing key pair, you export the key pair to a disk to store on the SAML extension server.

To export the key pair object:

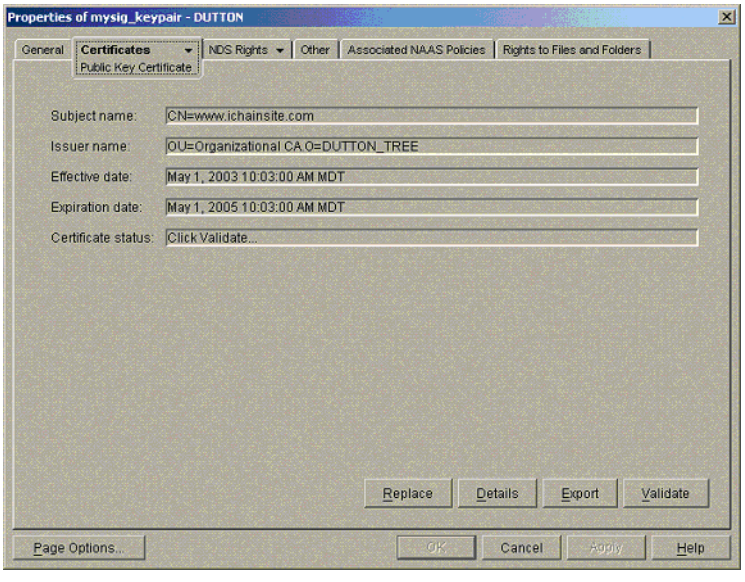
1 Double-click or right-click the key pair object and click Properties. This launches the key pair's property page, as shown in Figure 77:

Figure 77 Key Pair Properties



2 Click the Certificates tab, and select Public Key Certificate. A property page similar to Figure 78 is displayed:

Figure 78 Public Key Certificate Property Page

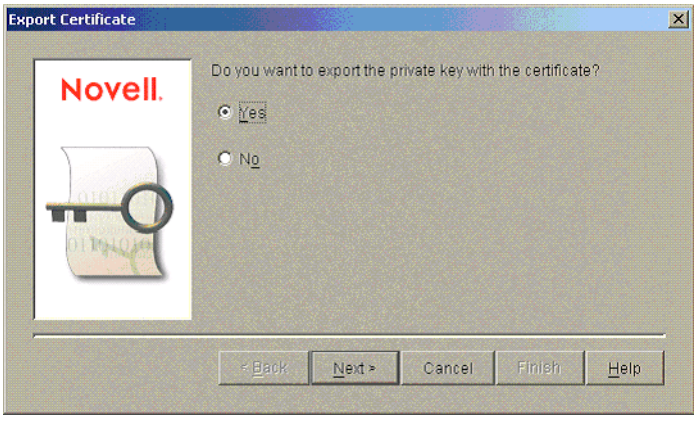


Verify that the Subject Name, Effective, and Expiration dates are the same as those you entered during the certificate creation process.

3 To export the key pair object, click Export.

A wizard page is displayed, as shown in Figure 79:

Figure 79 Key Pair Wizard Page



4 Because you are exporting the public-private key pair, leave the Yes default selected.

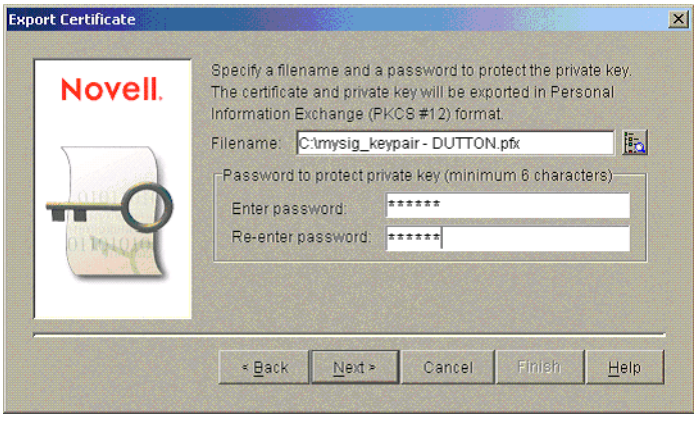
Selecting Yes exports the public and private key in password-protected PKCS#12 format. This is what you must move to the SAML extension server.

If you want to export only the public key certificate, select No.

Selecting No exports only the public key. You provide this to your public sites later.

5 Click Next.

Figure 80 Filename Wizard Page



6 Specify your desired filename. The password you enter on this page is important because it is required when you configure the SAML extension server to use the key pair.

7 Click Finish.

Setting the PKCS#12 Signature Key on the SAML Extension Server

To import the signature key file into the SAML extension server for use:

1 Copy the PKCS#12 file exported in the previous section (see "Exporting a Signing Key Pair" on page 77) to the local drive of the SAML extension server.

Ideally, this would be in the same directory as the samlextConfig.xml file.

2 Modify the SAML Extension server configuration file to point to this file.

When you installed SAML extension, a configuration file was automatically generated. This file is located at *samlext_home*/samlext_home/config/samlextConfig.xml. If no key pairs were specified during the install, the configuration file should look similar to Figure 81:

Figure 81 SAML Extension Configuration File

```
<authority name="SAMLExtDirectoryAuthority">
        <class><name>com.novell.wss.authority.saml.SAMLExtDirectoryA
uthority</name></class>
        <data>
                <servers>
                        <url>ldap://137.65.159.66:389</url>
                </servers>
                <proxyUser>
                        <dn>cn=admin,o=novell</dn>
                        <password>novell</password>
                </proxyUser>
                <isoDn>cn=iso,o=novell</isoDn>
                <initialCapacity>10</initialCapacity>
                <maxCapacity>30</maxCapacity>
                <keypairs>
                        <keypair usage="ssl" type="jks">
                                <password></password>
                                <file></file>
                        </keypair>
                        <keypair usage="signing" type="jks">
                                <password></password>
                                <file></file>
                        </keypair>
                </keypairs>
        </data>
</authority>
```

3 Modify the signature keypair element with usage signing to include the filename and password of the PKCS#12. In this example, the file is modified to read as shown in Figure 82:

Figure 82 Modified Configuration File

```
<authority name="SAMLExtDirectoryAuthority">
        <class><name>com.novell.wss.authority.saml.SAMLExtDirectoryA
uthority</name></class>
        <data>
                <servers>
                        <url>ldap://137.65.159.66:389</url>
                </servers>
                <proxyUser>
                        <dn>cn=admin,o=novell</dn>
                        <password>novell</password>
                </proxyUser>
                <isoDn>cn=iso,o=novell</isoDn>
                <initialCapacity>10</initialCapacity>
                <maxCapacity>30</maxCapacity>
                <keypairs>
                        <keypair usage="ssl" type="jks">
                                <password></password>
                                <file></file>
                        </keypair>
                        <keypair usage="signing" type="pkcs12">
                                <password>novell</password>
                                <file>c:\mysig_keypair.pfx</file>
                        </keypair>
                </keypairs>
        </data>
</authority>
```

In this example, it is assumed that you copied the exported PKCS#12 file to the SAML extension server as c:\mysig_keypair.pfx using novell as the password.

NOTE: After editing the samlextConfig.xml file, you must restart Tomcat.

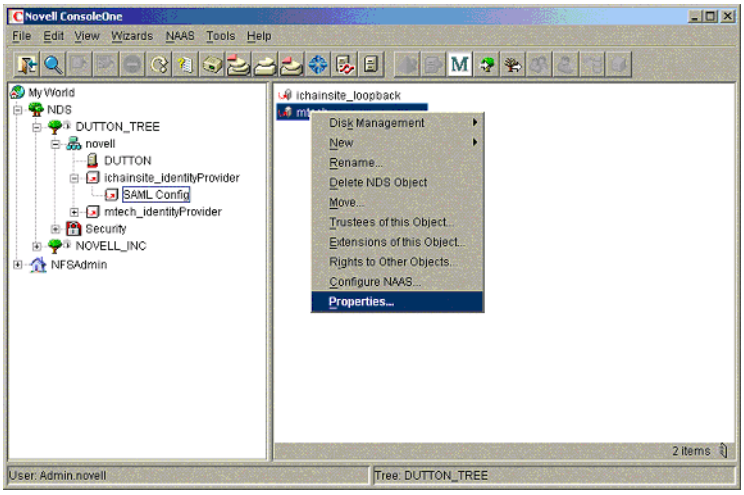
Modifying SAML Settings in the Directory

Even though you have created and imported the key pair, it is only used if it is required by the SAML configuration stored in the directory. The signing of SAML data is a setting made on a per-affiliate basis. This means that the SAML administrator can decide which SAML Trusted Affiliate sites receives signed data and which do not.

To modify the SAML settings in the directory:

1. Select the Trusted Affiliate object you want to sign data for.
2. Open the Trusted Affiliate object's property page, as shown in Figure 83:

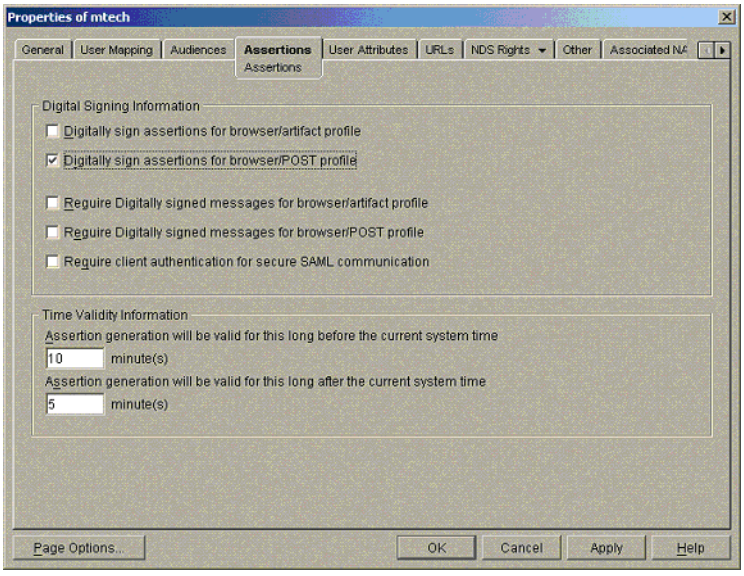
Figure 83 Trusted Affiliate Object Property Page



3 Select Assertions > Digitally sign assertions for the browser/Post profile.

This causes the system to use your key pair to sign SAML data sent to this Trusted Affiliate using the browser/POST profile. See Figure 84:

Figure 84 Digitally Sign Assertions for Browser/POST Profile



Exporting the Public Key Certificate

Now that you will be signing the SAML data, you need to provide your SAML Trusted Affiliate site with a way of validating the signatures you generate. You do this by providing the Trusted Affiliate with your public key certificate, which it can import into its system to use to validate signatures you generate.

To export the public key certificate:

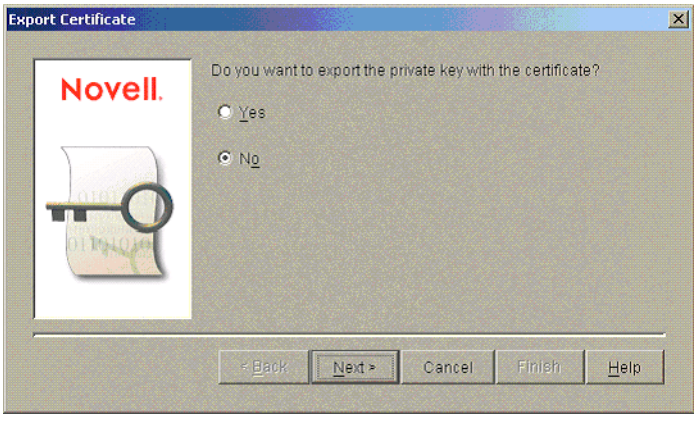
1 Open the Properties page associated with the key pair you are using to generate your digital signatures, the same as you did when you exported the key pair in PKCS#12 format.

2 Select Certificates > Public Key Certificate.

3 Click Export.

A wizard page is displayed, as shown in Figure 85:

Figure 85 Export Wizard Page

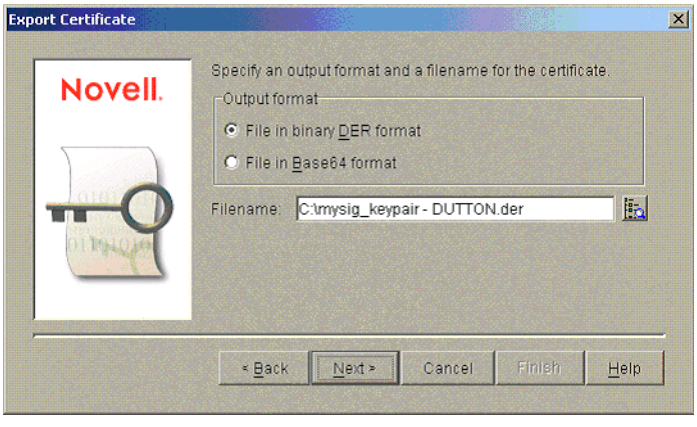


4 Select No to prevent exporting the private key with the certificate. This causes only the public key portion of the key pair to be exported.

5 Click Next.

6 Select the file name and format to save the file as. The most common file format is binary DER encoding. See Figure 86:

Figure 86 Exporting the Private Key With the Certificate



At this point, if you were to send out this public key certificate file to your partner sites that want signed data, the partner sites would then import the certificate so that they could validate your signatures.

IMPORTANT: Until the Trusted Affiliate receives and imports your public key certificate into its system, it cannot validate your signatures. Make sure that the Trusted Affiliate partner site has successfully imported and configured the certificate before attempting to send digitally signed assertions.

Validating Digital Signatures

This section covers the viewpoint of a site administrator who wants to receive signed SAML data from a Trusted Affiliate. At this point, you are configuring properties relating to the receiving half of the SAML relationship.

Assume that you have a SAML trust relationship with another site, and that site is digitally signing incoming SAML assertions to you. You must get a Public Key Certificate from the Trusted Affiliate site in order to validate the signatures it is going to be generating. This means you must:

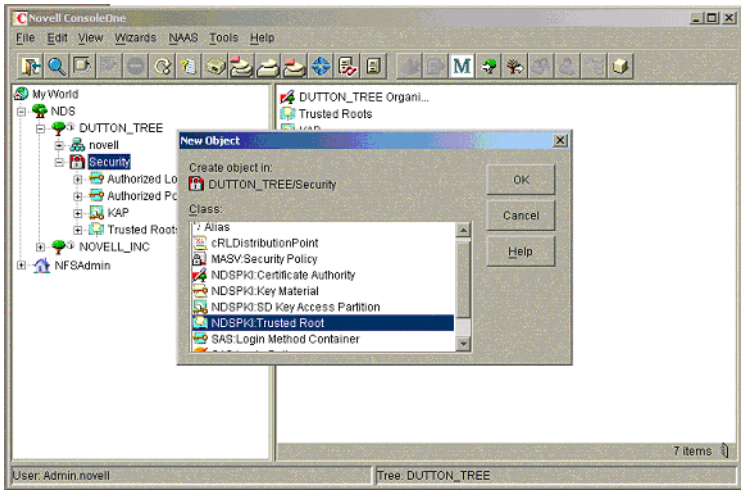
1. Receive a public key certificate from the Trusted Affiliate.
2. Import the public key certificate into eDirectory.
3. Associate the public key certificate with the appropriate Trusted Affiliate.
4. Set the SAML signature security settings on the appropriate Trusted Affiliate.

You can receive public key information in a number of different ways. The most common way is to receive a .der encoded certificate file through e-mail or on disk. After receiving the certificate file, you must import it into eDirectory so that it can be used.

In Novell eDirectory, trusted public key certificates must be placed in a Trusted Roots container. If you do not already have one, you need to create this container. Usually the Trusted Roots container is created under the [ROOT].security container. Do the following to create the container:

1. Right-click the security container.
2. Select New Object > NDSPKI:Trusted Root, as shown in Figure 87:

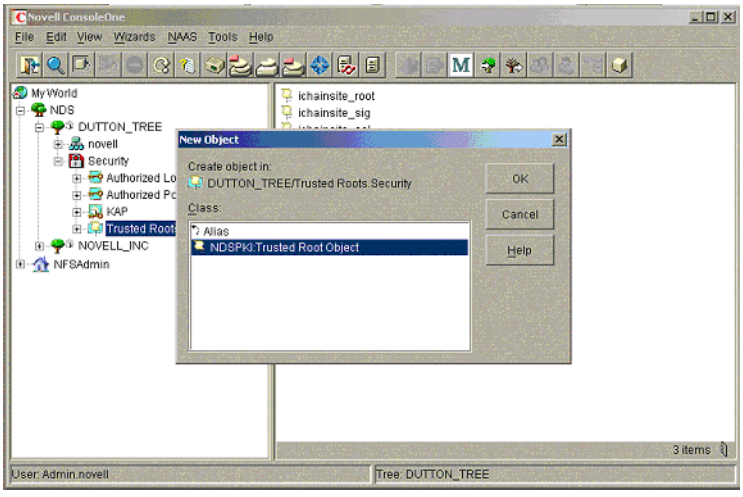
Figure 87 NDSPKI:Trusted Root



3. Click OK.
4. Add a Trusted Root public key to the newly created Trusted Roots container. Right-click the Trusted Roots container, and select New > Trusted Root.

   The Create Trusted Root Wizard launches, as shown in Figure 88:

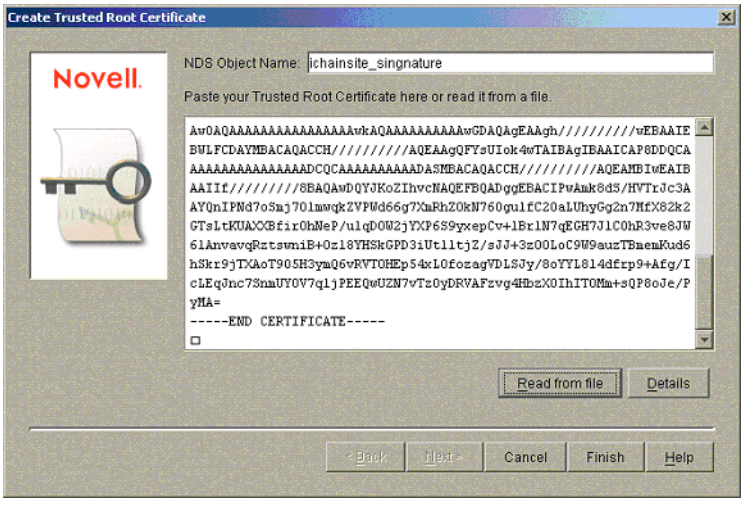
Figure 88 Create Trusted Root Wizard



5 Enter the name for the Trusted Root object, then import the file.

A panel similar to the following should be displayed.

Figure 89 Trusted Root Page



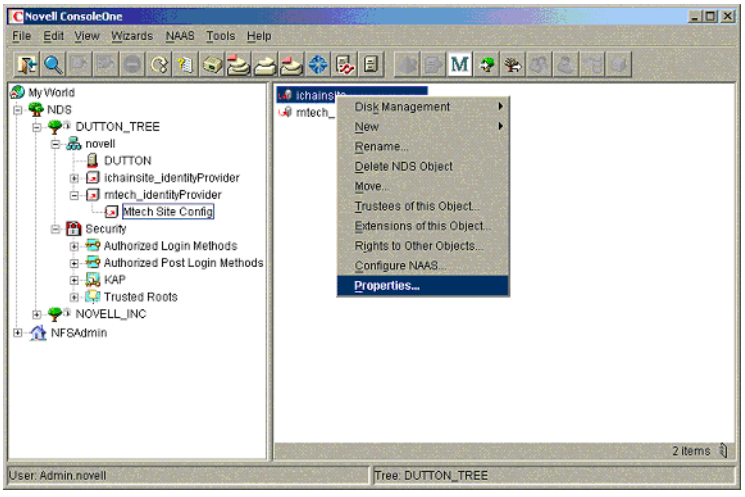
6 Click Finish to complete the import.

You should now have a new object in the Trusted Roots container.

Next, you must associate this object with your Trusted Affiliate entry for the appropriate Trusted Affiliate object.

7 Right-click the Trusted Affiliate object, then select Properties, as shown in Figure 90.

Figure 90 Trusted Affiliate Properties

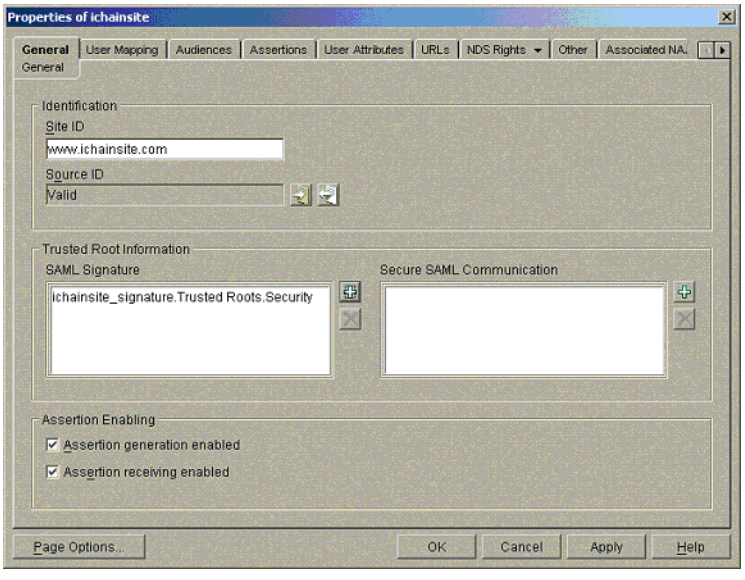


8 Click the General tab, then click the plus sign (+) located under Trusted Root Information > SAML Signature.

9 Browse to the Trusted Roots container and select the desired Trusted Root object.

Figure 91 shows a Trusted Affiliate with a SAML Signature Public Key Certificate set:

Figure 91 Trusted Affiliate With SAML Signature Public Key Certificate Set

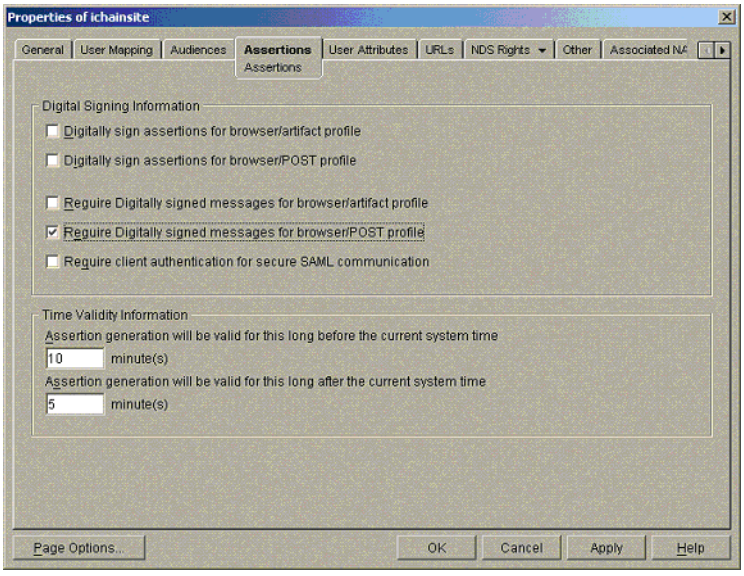


This setting allows the SAML extension server to validate digital signatures generated by this Trusted Affiliate using the specified certificates.

10 Finally, you can require that all SAML data sent from this Trusted Root must be digitally signed. Click the Assertions tab, then select the Require Digitally Signed Messages for the Browser/POST or Require Digitally Signed Messages for the Browser/Artifact check box.

According to the SAML specification, while requiring browser/POST signatures, browser/Artifact signatures are optional, but browser/POST signatures are required. See Figure 92:

Figure 92 Assertions Tab: Signatures



SAML Secure Communication: SSL - Mutual

In the SAML browser/Artifact profile, SAML partner sites directly communicate in order to pass SAML assertions back and forth. This direct server-to-server communication is often referred to as the SAML back-channel. SAML back-channel communication should always go over SSL. For added security, the SAML browser/Artifact profile requires that the SSL layer be mutually authenticated. This means that both the server and the client must mutually authenticate using certificates.

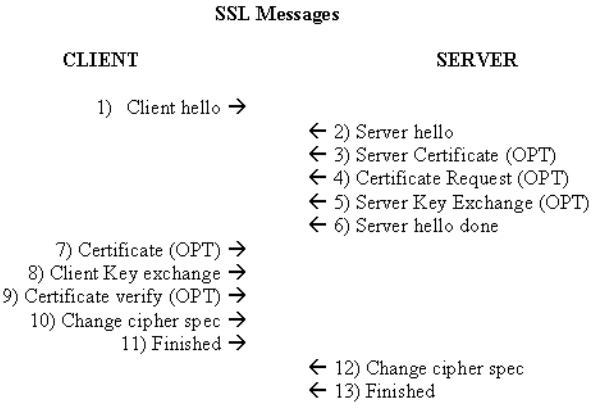
The following are commonly used terms when discussing SSL and Public Key Infrastructure (PKI) in general:

- **Trusted Root Certificate**: Certificates are generally signed by some parent Certificate Authority (CA). The idea is that if you trust a given CA, then you automatically trust the certificates that it signed.
- **Trusted Certificate:** Instead of importing a trusted CA that causes you to trust all of the certificates signed by that CA, you can import an individual public key certificate that you want to trust. This way, you are assured that you will trust that certificate only, as opposed to everything signed by a given CA.
- **SSL Server Certificate:** The certificate used by an SSL server to authenticate the server to incoming requests
- **SSL Client Certificate:** A certificate used to authenticate to servers when generating outbound requests.

SSL Flow of Messages

Figure 93 shows the flow of messages that occur between a Client and Server in setting up an SSL connection:

Figure 93 SSL Message Flow



The SSL message flow shows the two critical phases of the SSL negotiation where trust is established between the Server and Client:

1. In Step 3, the server presents its SSL Server Certificate. This certificate is used by the Client in Step 9 to determine if the server is trusted or not. These steps authenticate the Server to the Client.
2. If client authentication is performed (SSL - Mutual), the server requests a Client certificate (Step 4), and the Client sends its SSL Client Certificate to the Server, which is then responsible for validating it. This step authenticates the Client to the Server.

Determining Whether to Trust a Certificate

- Certificate Trust in iChain
- Certificate Trust In the SAML Extension Server

Certificate Trust in iChain

iChain determines trust using its Trusted Roots container. If the provided SSL Client Certificate exists in its Trusted Roots container, iChain trusts the certificate; or, if the provided SSL Client Certificate was signed by a CA that is in the Trusted Roots container, then iChain trusts the certificate. If neither the client certificate nor its CA is found in iChain's Trusted Roots container, the SSL Client certificate is rejected and the connection closed.

Certificate Trust In the SAML Extension Server

The SAML extension server determines which certificates it trusts based upon the settings in each Trusted Affiliate object. The Trusted Affiliate's General page contains a Trusted Root Information section for the UI. This panel allows the SAML Site administrator to specify which SSL certificates are trusted by the system. When the SAML extension server loads its configuration from the directory, each of the Trusted Affiliate's Secure SAML Communication certificates are read. These certificates are added to the Java Virtual Machine's (JVM) Trust Store. When the

SAML extension server generates an outbound SSL request, the received SSL Server Certificate is checked against the certificates stored in the JVM. The certificate is trusted and a connection is made if:

- The SSL Server Certificate exists in the JVM Trust Store.
- The SSL Server Certificate's CA exists in the JVM Trust Store.

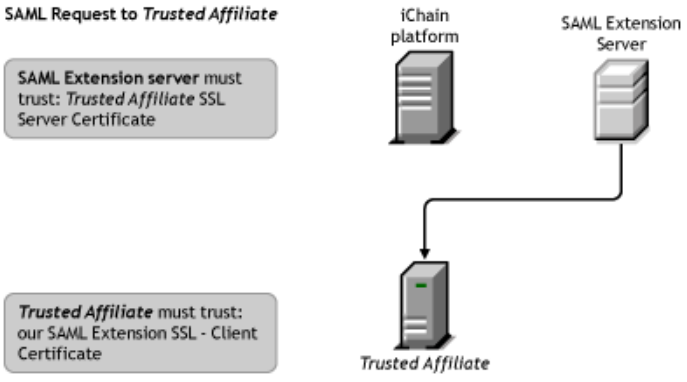
The next step in understanding how SAML uses SSL is to look at how the SAML messages are sent between the SAML Extension server and the Trusted Affiliate sites.

Sending SAML Requests

Communication between SAML partner sites involves both generating outbound SAML Requests to partner sites and responding to incoming SAML Requests from partner sites.

The SAML extension server can send SAML Requests to partner sites. This occurs in the SAML browser/Artifact profile when an incoming user from a Trusted Affiliate attempts to authenticate using a SAML Artifact. The SAML extension server must request the corresponding SAML assertion from the referring SAML Trusted Affiliate. This request can be made over mutually authenticated SSL. Figure 94 shows the required interactions for this communication to occur:

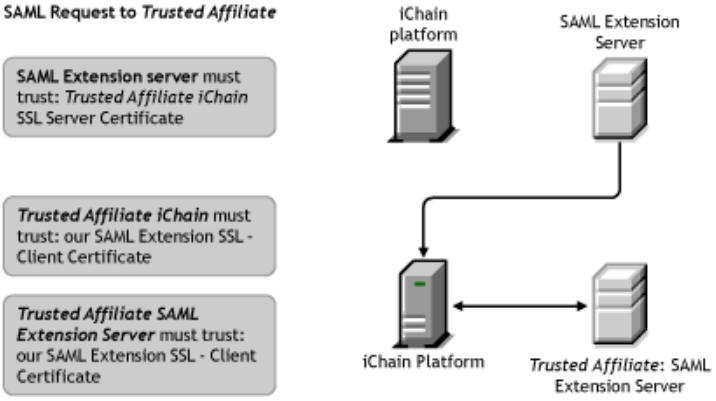
Figure 94 Required Interactions



In order for the communication to occur, the SAML extension server must have in its Trust Store the SSL Server Certificate of the Trusted Affiliate to whom the request is sent. If SSL with mutual authentication is being used, the Trusted Affiliate must have the SAML extension's SSL Client Certificate in its Trust Store.

If the Trusted Affiliate in this example were an iChain site, the interaction would be as shown in Figure 95:

Figure 95 iChain site as the Trusted Affiliate

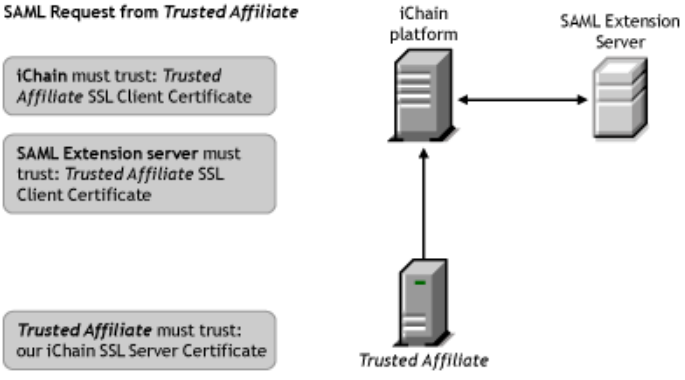


All traffic intended for the SAML extension server passes through iChain first. Thus, incoming requests creates SSL connections with iChain rather directly with the SAML extension server. This means that in order to establish trust, the Trusted Affiliate site must have the iChain SSL Server certificate in its Trust Store.

Receiving SAML Requests

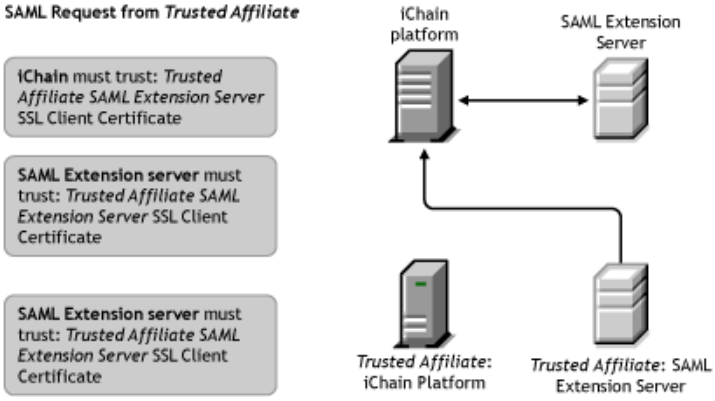
The SAML Extension for iChain must respond to SAML requests from Trusted Affiliate sites. This is commonly done when the SAML extension server has sent an outbound user to a Trusted Affiliate site using the SAML browser/Artifact profile. In this case, the Trusted Affiliate site must send a SAML Request to your site to request the SAML assertion corresponding to the provided SAML Artifact. All traffic intended for the SAML extension server must first pass through the iChain proxy. Figure 96 shows the communication details:

Figure 96 Communication Pattern



In order for trust to be established in this case, the Trusted Affiliate site must trust the iChain SSL-Server certificate and, if SSL mutual authentication is desired, the Trusted Affiliate SSL Client Certificate must be trusted by iChain. If the Trusted Affiliate site were running the SAML extension for Novell iChain, Figure 97 would apply:

Figure 97 Trusted Affiliate Running SAML Extension



Since outbound SAML requests are sent directly from the SAML extension server, this example does not differ much from the previous non-iChain scenario. The iChain platform must trust the SSL Client of the Trusted Affiliate SAML extension server, and if SSL mutual authentication is being used, iChain must trust the SSL Client certificate in use by the Trusted Affiliate SAML extension server.

SSL Trust Configuration

In order for mutually authenticated communication to occur between two SAML sites, you must have the following:

- The sites must trust each other's SSL Client certificates.
- The sites must trust each other's SSL Server certificates.

For the SAML extension for Novell iChain, this is done by adding trust for the Trusted Affiliate Site:

1. Import the Trusted Affiliate site's SSL Server Certificate into the iChain Trusted Roots container.
2. Import the Trusted Affiliate site's SSL Client Certificate into the iChain Trusted Roots container.
3. Add a reference in SAML Configuration Trusted Affiliate > General > Secure SAML Communication to the imported SSL Server Certificate.
4. Add a reference in the SAML Configuration Trusted Affiliate > General > Secure SAML Communication to the imported SSL Client Certificate.

To allow the Trusted Affiliate to trust you:

1. Export the public key certificate associated with your iChain SSL Server Certificate.
2. Export the public key certificate associated with our SAML extension server SSL Client Certificate.
3. Send the two certificates to the Trusted Affiliate.

After you understand the trust relationship that must exist between your site and your Trusted Affiliate partners, you can set up the system for SSL and SSL with mutual authentication.

Setting Up SSL

The following steps must be followed in order to configure the SAML system support SSL mutual authentication over the SAML back-channel.

1. An SSL server key pair (SKP) must be obtained and imported into eDirectory.
2. An SSL client key pair (CKP) must be obtained and imported into eDirectory.
3. The SKP must be made available in PKCS#12 format and imported into the iChain server.
4. The SKP must be configured for use as the SSL server key pair on the appropriate iChain accelerator.
5. The CKP must be made available to the SAML extension server in PCKS#12 or Java Key Store (JKS) format.
6. The SAML extension server must be configured to use the provided CKP as its SSL Client certificate.
7. The public key certificate associated with the SKP must be exported and sent the Trusted Affiliates.
8. The public key certificate associated with the CKP must be exported and sent to the Trusted Affiliates.
9. Your partner site must send you its SSL public key certificate, which you must import into your trust store.
10. The appropriate settings must be made on the Trusted Affiliate who will be communicating with your site over SSL-M (mutual).

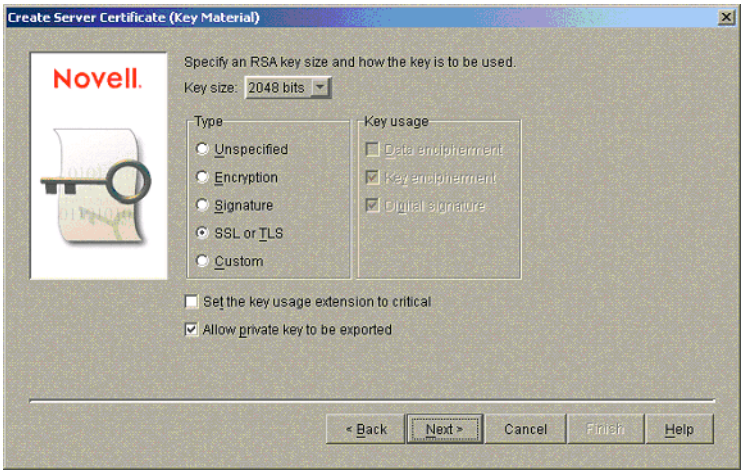
SSL Key Pair Generation

You can use the Novell Certificate Server snap-ins to generate your SSL key pair. If you choose to do this, the steps required are nearly identical to those followed to generate the data signing key. The only difference is on the Create Server Certificate page, rather than selecting the Signature option as you did when you were generating the signature key pair, you should select the SSL or TLS button, as shown in Figure 98.

See “Creating a Signing Key Pair” on page 72 for the steps to generate the data signing key.

Also, when you are generating the SSL Server Certificate, the Subject Name of the certificate must match the host name of the server. For instance, if you were generating an SSL Server certificate for www.novell.com, the CN in the Subject Name on the certificate must be .CN=www.novell.com.

Figure 98 SSL or TLS Button



Exporting the SSL Key

The steps for exporting the SSL key are identical to those used to export the signing key pair. (See "Exporting a Signing Key Pair" on page 77). You must select the Public Key Certificate tab and remember the export password, because you will need it when the key pair is imported.

Importing the SSL Key Into iChain

To import the SSL key into iChain:

1. If you already have a good certificate in use on the iChain accelerator, you can use the iChain GUI to export it to a PKCS#12 file, and then import it into eDirectory.

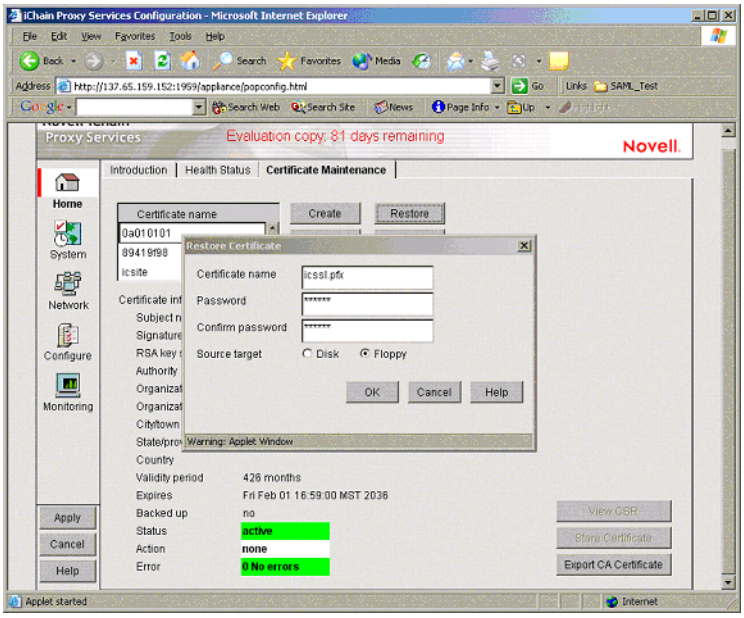
   Alternately, you can import a PKCS#12 key pair exported from eDirectory into iChain.

   NOTE: In order to import a PKCS#12 file into iChain, the file must be in 8.3 format. That is, the file must not have more than eight characters in the name, and it must have a three-character extension. Copy the PKCS#12 file you exported onto a floppy and rename it to fit the 8.3 format, if necessary.

2. Open the iChain GUI on the iChain server.

3. Select the Certificate Maintenance option and click the Restore button. You should see a screen similar to Figure 99:

Figure 99 Restore Certificate

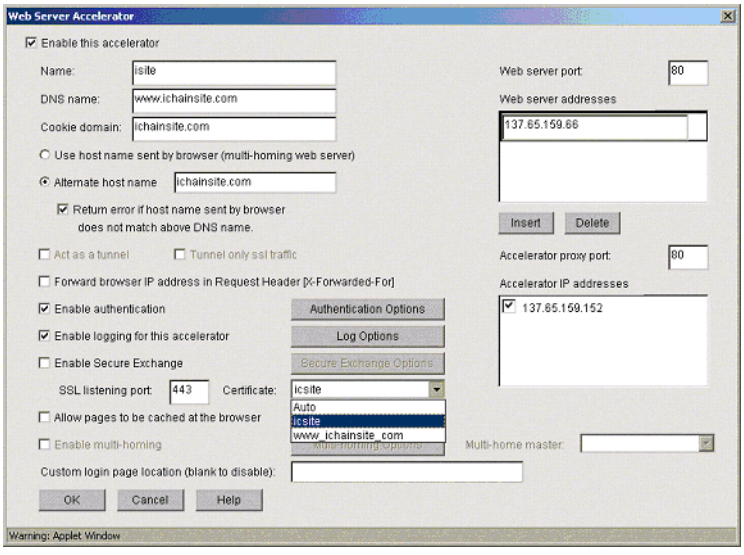


4 Enter the PKCS#12 filename, enter the appropriate password, and select the Floppy option.

After selecting OK, click the Apply button on the left side of the GUI. The status and action indicators on the Certificate Maintenance page show whether the operation was successful.

5 Next, you need to configure the appropriate iChain accelerator to use this certificate for SSL. Select Configure > Web Server Accelerator. Select the appropriate accelerator, then click Modify. You should see a page similar to Figure 100:

Figure 100 Web Server Accelerator



6 Use the Certificate drop-down menu to select the certificate you want imported or generated, then apply the changes you made.

Importing and Configuring the SAML Extension Server To Use the SSL Certificate

As you did in the signing key case, you must get the PKCS#12 file exported from eDirectory onto the local file system of the SAML extension server, then modify the SAML extension server's configuration file to use it.

To import the Signature Key file into the SAML extension server for use:

1. Copy the PKCS#12 file exported in the previous process to the local drive of the SAML extension server.
2. Modify the SAML extension server configuration file to point to this file.

When you installed the SAML extension, a configuration file was automatically generated. This file is located at *samlext_home*/config/samlextConfig.xml. After modifying this file to handle the signing key, it should look like the Figure 101:

Figure 101 File Modification for the Signing Key

```
<authority name="SAMLExtDirectoryAuthority">
        <class><name>com.novell.wss.authority.saml.SAMLExtDirectoryA
uthority</name></class>
        <data>
                <servers>
                        <url>ldap://137.65.159.66:389</url>
                </servers>
                <proxyUser>
                        <dn>cn=admin,o=novell</dn>
                        <password>novell</password>
                </proxyUser>
                <isoDn>cn=iso,o=novell</isoDn>
                <initialCapacity>10</initialCapacity>
                <maxCapacity>30</maxCapacity>
                <keypairs>
                        <keypair usage="ssl" type="jks">
                                <password></password>
                                <file></file>
                        </keypair>
                        <keypair usage="signing" type="pkcs12">
                                <password>novell</password>
                                <file>c:\mysig_keypair.pfx</file>
                        </keypair>
                </keypairs>
        </data>
</authority>
```

Modify the signature keypair element with usage SSL to include the filename and password of the SSL key pair PKCS#12. In this example, you would modify the file to read as shown in Figure 102:

Figure 102 Modifying the Signature Keypair Element

```
<authority name="SAMLExtDirectoryAuthority">
        <class><name>com.novell.wss.authority.saml.SAMLExtDirectoryA
uthority</name></class>
        <data>
                <servers>
                        <url>ldap://137.65.159.66:389</url>
                </servers>
                <proxyUser>
                        <dn>cn=admin,o=novell</dn>
                        <password>novell</password>
                </proxyUser>
                <isoDn>cn=iso,o=novell</isoDn>
                <initialCapacity>10</initialCapacity>
                <maxCapacity>30</maxCapacity>
                <keypairs>
                        <keypair usage="ssl" type="jks">
                                <password>novell</password>
                                <file>c:\myssl_keypair.pfx</file>
                        </keypair>
                        <keypair usage="signing" type="pkcs12">
                                <password>novell</password>
                                <file>c:\mysig_keypair.pfx</file>
                        </keypair>
                </keypairs>
        </data>
</authority>
```

This assumes that you copied the exported PKCS#12 file to the SAML extension server as c:\myssl_keypair.pfx using a password of novell.

Exporting the SSL Public Key Certificate

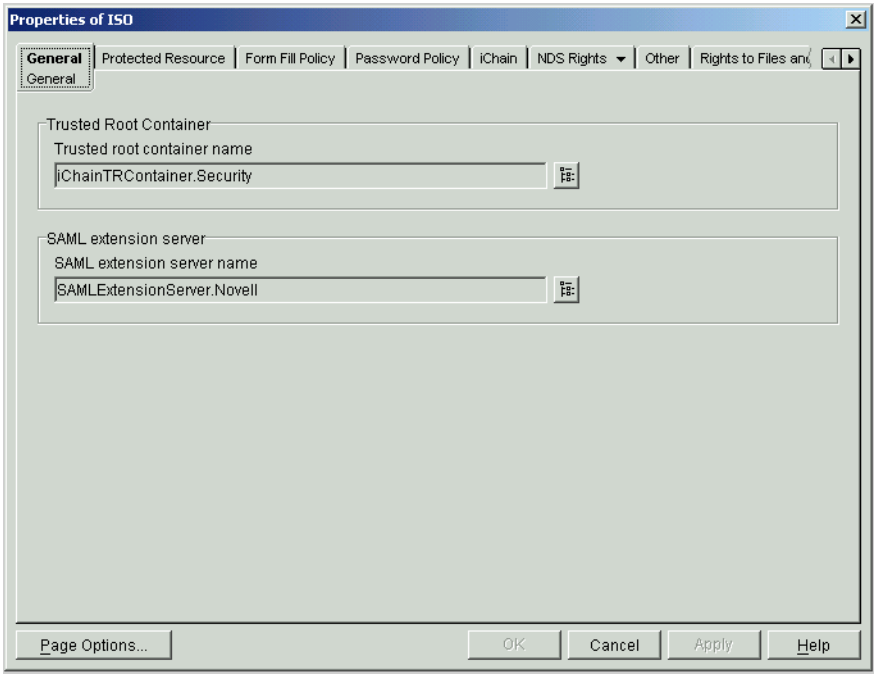
In order for your partners to accept SSL connections from you, they must have and trust the public key associated with your SSL key pairs. You must export the public key certificates associated with the SSL Server and SSL Client certificates and send them to your Trusted Affiliate sites. SSL public key certificates are exported in the same way you exported the public key for your signing key pair.

Importing the Partner SSL Public Key Certificate

In order to create SSL connections with your Trusted Affiliate partner sites, you must import your SSL Server and SSL Client public key certificates into the iChain and Trusted Affiliate objects.

In order to import these certificates, you must first determine the name of the Trusted Roots container that iChain is using to hold its trusted roots certificate in. This value is determined by opening the Properties page of the iChainServiceObject. The Trusted Roots Container value on the General page shows the location of the appropriate Trusted Roots Container, as shown in Figure 103:

Figure 103 Properties of the iChainServiceObject

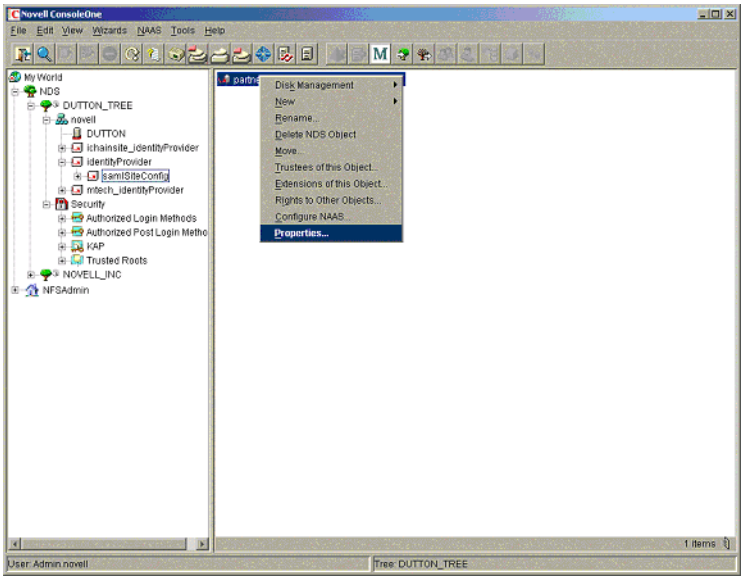


If the value has not yet been set; you can set it by selecting the Browse button on the right. If no Trusted Roots Container has been created, follow the steps outlined in “Generating SAML Digital Signatures” on page 71 to create one.

After you set the Trusted Roots Container name attribute in the iChainServiceObject, browse to the container and import the certificate, as outlined in the “Generating SAML Digital Signatures” on page 71.

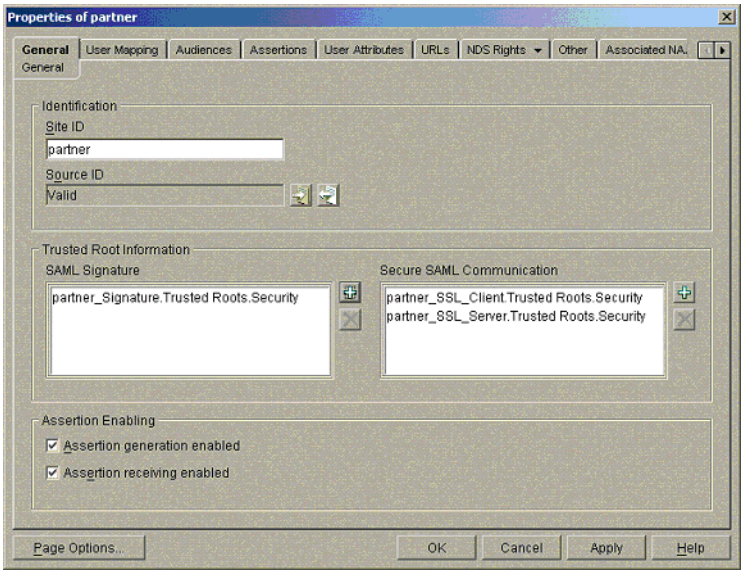
After importing the certificates into the appropriate Trusted Roots Container, you must configure the SAML extension server to use them. Access the appropriate Trusted Affiliate object (this is the Trusted Affiliate object associated with the certificates) and open its Properties page.

Figure 104 Trusted Affiliate Object Properties



Select the General tab. Click the plus sign (+) button in the Secure SAML Communication group. Browse to the appropriate Trusted Roots Container and select the SSL certificates associated with this Trusted Affiliate.

Figure 105 SSL Certificates Associated With the Trusted Affiliate



This example shows that two certificates have been associated with the Trusted Affiliate named partner. They are partner_SSL_Client.Trusted Roots.Security and partner_SSL_Server.Trusted Roots.Security. These two certificates are trusted by both the iChain server and SAML extension server for creating SSL connections.

Modifying the SAML SOAP Endpoint URL

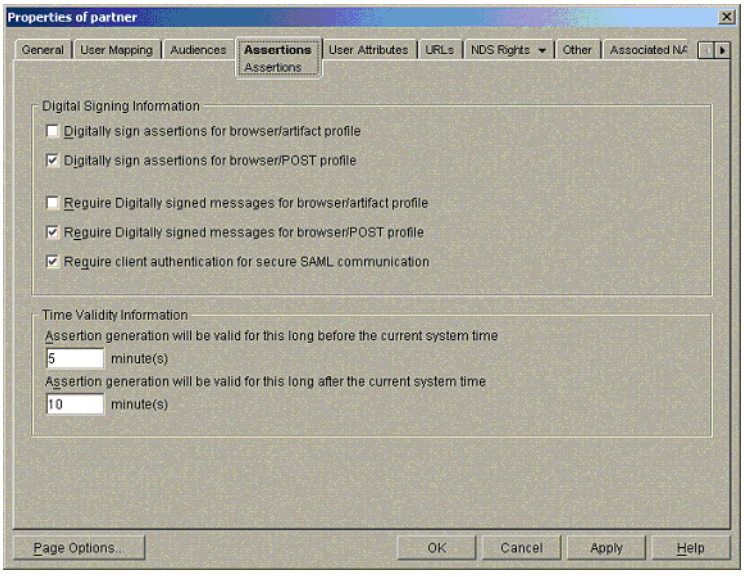
The SAML extension server can be accessed using the following two URL extensions:

- **http(s)://*host*/cmd/ext/samlext/saml/resp:** Used for non-mutually authenticated SSL connections.
- **https://*host*/cmd/mutExt/samlext/saml/resp:** Used for SSL mutual connections only.

If you want Trusted Affiliate partner sites to access your site using only SSL with mutual authentication, they must use the second URL (/cmd/mutExt).

You can require that a given Trusted Affiliate use SSL mutual by modifying the settings on the Assertions page of the specified Trusted Affiliate's Properties page, as shown in Figure 106:

Figure 106 Trusted Affiliate Properties Page



If the Require client authentication for secure SAML communication options is selected, only communication over (/cmd/mutExt) is accepted by the system.

The SOAP Responder URL now contains /cmd/mutExt, rather than /cmd/ext. You can require that affiliates communicating with you over the SAML back-channel use SSL-M. This setting is made on the Assertions page.

With the Require Client Authentication for Secure SAML Communication setting enabled, only connections with SSL-M and with a certificate matching that in the Secure SAML Communication field are accepted.

6 Enabling Web Sites With SAML Single Sign-On Functionality

This section discusses what back-end Web site developers must do in order to utilize SAML single sign-on functionality provided in the SAML extension for Novell® iChain®.

A SAML single sign-on transaction has a referring site, where the user originally authenticated, and a receiving site, which is a partner site the user wants to access. The referring site is responsible for creating a SAML assertion on behalf of the user and redirecting that user to the appropriate SAML receiving resource on the receiving site. The receiving site is responsible for accepting the SAML assertion, authenticating the user, and providing the user with the target resource. A Web site developer can use the SAML extension for iChain to act as both a referring and receiving site.

This section discusses the following topics:

Acting As a Referring Site

The SAML referring site is responsible for generating a SAML single sign-on assertion for the user and sending that user to the receiving site. The goal of this action is that the user is authenticated and granted access to some resource at the receiving site. In order to do this, SAML defines a resource called an Intersite Transfer URL. This resource is provided by the SAML Extension for Novell iChain. It is accessed via the following URLs:

- **http(s)://*host*/cmd/ext/samlext/saml/gen/post:** Generates SAML assertions in the Browser/POST profile.
- **http://*host*/cmd/ext/samlext/saml/gen/afct:** Generates SAML assertions in the Browser/Artifact profile.

The purpose of the Intersite Transfer service is to generate SAML assertions for users and redirect them to the receiving site. In order for the Intersite Transfer service to do this, it needs two pieces of information:

- **AID:** The Affiliate ID or Site ID of the receiving site. This value allows the SAML Intersite Transfer to create the proper type of SAML assertion for the partner site. This value must match the Site ID of the desired Trusted Affiliate configuration object in the directory.
- **TARGET:** The resource the user wants to access at the receiving site.

These two pieces of information are provided to the Intersite Transfer service as URL parameters. Thus, a complete Intersite Transfer link would look like the following:

- **Http(s)://*host*/cmd/ext/samlext/saml/gen/post?AID=partner_site&TARGET=https://www.partner_site.com/resource.html:** Generates a SAML assertion for Trusted Affiliate

partner_site and request the resource https://www.partner_site.com/resource.html. The SAML Browser/POST profile is used.

- **http(s)://*host*/cmd/ext/samlext/saml/gen/afct?AID=partner_site&TARGET=https://www.partner_site.com/resource.html:** Generates a SAML assertion for Trusted Affiliate partner_site and request the resource https://www.partner_site.com/resource.html. The SAML Browser/Artifact profile is used.

To further illustrate how the SAML Inter-site Transfer service works, consider the following example: You are developing a travel site (travel4cheap), and you have partnered with a car rental company named cars4rent. You have already created a Trusted Affiliate object in the directory for cars4rent, and have set the Site ID to cars4rent. The resource at your partner site that you want to give your customers access to is http://www.cars4rent.com/partners/rentacar.html. Your goal is to have the travel site provide users with a link that sends them to cars4rent and provides them with access to the target resource without needing to re-authenticate at the cars4rent site.

Suppose that in the past a direct link was in place to the cars4rent resource that, when followed, required the users to authenticate to cars4rent. The following is the original HTML:

```
…
<A href="https://www.cars4rent.com/deals.html">Access Partner Cars4Rent</A>
…
```

To use the SAML Extension for Novell iChain, replace the original link with the following:

```
…
<A href="https://www.travel4cheap.com/cmd/ext/samlext/saml/gen/
post?AID=cars4rent&TARGET= https://www.cars4rent.com/deals.html">Access
Partner Cars4Rent</A>
…
```

The above link will invoke the SAML Intersite Transfer server to generate a SAML single sign-on assertion for the user intended for the cars4rent partner site. The users can then access the cars4rent target resource without re-authenticating to cars4rent. The important differences between the original and SAML enabled links are:

- The link now points to travel4cheap Intersite Transfer URL rather than directly to cars4rent
- An AID parameter with the value cars4rent was added.
- A TARGET parameter with the value https://www.cars4rent.com/deals.html was added.

NOTE: The order of AID and TARGET is immaterial.

In order for this scenario to work, the SAML Configuration must contain a Trusted Affiliate object with Site ID cars4rent, and the cars4rent partner site must have a SAML service running to receive the SAML assertion.

No other changes are required on the back-end Web site. All that needs to be done in order to act as a SAML referring site is to create links to the SAML Intersite Transfer URL with the proper parameters. No other Web site changes are required.

Sending User Attributes to Partner Sites

In some cases, receiving sites might want to know more about incoming users than simply their username and authentication information. The SAML specification allows user attribute information to be shared between partner sites. The SAML Extension for Novell iChain allows the administrator to specify any LDAP attribute to be included in generated SAML assertions. You configure for this on the properties page for the Trusted Affiliate object in the directory. The User

Attributes tab contains a list of attributes that can be made available to partner sites. For details on how to configure the system to send user attributes to Trusted Affiliates, see Chapter 3, "Configuring the SAML Extension," on page 39.

SAML Security Settings

Different SAML partnerships might have different security requirements. Some partnerships might have a relatively low value where the receiving site is simply offering the referring site's users a small discount or service. Other relationships could have high values where sensitive user attribute information is passed, such as a purchasing system where large purchases are made and user information such as credit card numbers are shared. The SAML configuration allows the administrator to determine what types of security constraints are applied to generated assertions The options are:

Sign Assertions in the Browser/POST Profile: This option generates a digital signature on assertions generated for the specified Trusted Affiliate using the Browser/POST profile. The inclusion of a digital signature allows the receiving site to validate the source of the provided assertions. Assertions generated under the Browser/POST profile must be signed to comply with the SAML specification version 1.0.

Sign Assertions in the Browser/Artifact profile: This option generates a digital signature on assertions generated for the specified Trusted Affiliate using the Browser/Artifact profile. According to the SAML specification, signatures for the Browser/Artifact profile are not mandatory; however, some partner sites might require them.

These settings are applied to the Trusted Affiliate object in the directory. The settings are located on the Trusted Affiliate properties page, under the Assertions tab. See Chapter 3, "Configuring the SAML Extension," on page 39 for more information.

Acting As a Receiving Site

Acceptance of SAML assertions and authentication of incoming users is all handed by the SAML extension service. The back-end Web server developer does not need to do anything different in order to support incoming SAML users. However, there are some issues to consider if OLAC is being used.

Using SAML With OLAC

OLAC allows back-end Web servers to receive user attribute information pulled from various data sources as HTTP header or query string parameters. As mentioned in "Sending User Attributes to Partner Sites" on page 102, user attribute information can be shared between SAML affiliate sites. It is possible to make the SAML attribute information available to back-end Web servers using OLAC. This is done by creating an entry in the OLAC table for a given protected resource with the DataSource = SAML. The value name should be the SAML Attribute name of the incoming SAML attribute. The Name value can be the same as that of other OLAC attributes. If other OLAC attributes exist with the same name, they are overridden if SAML OLAC data is available. This allows you to define protected resources that contain customized content that behave in the same way if the user authenticated directly to the system or was authenticated via a SAML assertion. See "Accessing SAML Attributes in OLAC" on page 64 for more information.

Security Constraints

Based upon the value of the partnership between two SAML partners, various levels of security might be required. The Trusted Affiliate configuration allows the administrator to determine what types of security constraints must exist in order for the system to accept SAML data from a specified Trusted Affiliate. The available options are:

Require Signatures on Browser/POST: This setting allows the administrator to require that SAML assertions sent in the Browser/POST profile contain a verifiable digital signature. This gives the receiving site the ability to verify that the provided assertion was indeed issued by the proper referring site. The SAML specification requires that SAML data sent in the Browser/POST profile contain a digital signature.

Require Signatures on Browser/Artifact: This setting allows the administrator to require that SAML assertions sent in the Browser/Artifact profile contain a verifiable digital signature. This gives the receiving site the ability to verify that the provided assertion was issued by the proper referring site. This is in addition to the security provided by the SAML back-channel communications layer associated with the SAML Browser/Artifact profile. The SAML specification does not require that Browser/Artifact assertions be signed; however, some partner sites might require it.

Require Mutual SSL: This setting requires that communication from this Trusted Affiliate be made over mutually authenticated SSL. According to the SAML specification, communication over the HTTPS back-channel must be mutually authenticated.

For more information on these settings, see Chapter 3, “Configuring the SAML Extension,” on page 39.

A Installing the Java Development Kit

The SAML extension server requires Java Development Kit (JDK) 1.4.1 or later. You can obtain the JDK from the Sun Java Web site (http://java.sun.com).

Downloading the JDK

To download the JDK:

1. At the main page of the Web site, click Java Standard Edition J2SE.
2. Near the top of the page, click J2SE Downloads.
3. At the download page, click the link to the J2SE 1.4.1 full download.
4. Download the installer, then run it.

   The installer will not create the required JAVA_HOME environment variable.You will need to manually set this variable.

B Installing Tomcat Light Edition

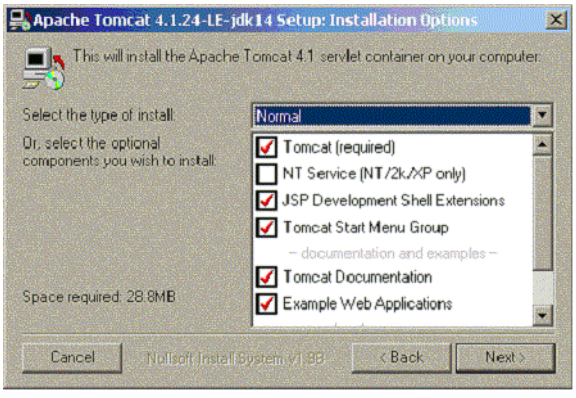
This appendix contains information about using Tomcat Light Edition (Tomcat LE).

Installing Tomcat LE

1. Download Tomcat LE from the Apache Web site (http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.1.24/bin).
2. Click jakarta-tomcat-4.1.24-LE-jdk14.exe. This is the download of the Windows installer for the LE edition.
3. Run the installer.

   The installation options dialog is displayed.

Figure 107 Installation Options



   Leaving the NT Service option unchecked is useful for debugging purposes, however, once you have completed the setup, you will want to have this service enabled so that it will start when the system boots. You also would not have to have a user logged in for it to run.

4. Click Next.
5. Select the destination directory for Tomcat.

   You might be prompted for admin information as part of the installation. This information will not affect the SAML extension server.

   You might also be prompted to enter a port for Tomcat to use. The standard Web server port is 80. By default, Tomcat uses 8080 to avoid conflicts with existing Web servers. Generally, it is easier to access the server if you use the standard port 80 for Tomcat.

Validating the Tomcat Installation

The Tomcat Start Menu Group creates an entry that allows you to start and shut down Tomcat. Alternatively, you can start Tomcat by opening a command prompt to the *TOMCAT_HOME*\bin directory and entering **`catalina run.`**

You can also enter the command **`catalina`** to view the various Tomcat startup options.

To verify that you successfully installed Tomcat:

1. Start Tomcat.
2. From a browser, enter http://localhost/ or http://localhost:8080.

   The Tomcat server should respond, indicating a successful installation.

For more information about installing and configuring Tomcat, see the Apache Tomcat project page (http://jakarta.apache.org/tomcat/index.html).

C Documentation Updates

This section lists updates to the *Novell® SAML Extension for Novell iChain® Administration Guide* that have been made since the initial release of SAML. The information will help you to keep current on documentation updates and, in some cases, software updates (such as a Support Pack release).

The information is grouped according to the date when the *Novell SAML Extension for Novell iChain Administration Guide* was republished. Within each dated section, the updates are listed by the names of the main table of contents sections.

The *Novell SAML Extension for Novell iChain Administration Guide* has been updated on the following dates:

January 26, 2005 (SP2)

Minor style and consistency changes throughout the guide for Support Pack 2.