CA CloudMinder™

SSO Partnership Federation Guide

1.53

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

CA Technologies Product References

This document references the following CA Technologies products:

- CA CloudMinder™ Identity Management
- CA CloudMinder™ Advanced Authentication
- CA CloudMinder™ Single Sign-On
- CA Directory
- CA IdentityMinder™
- CA AuthMinder™
- CA RiskMinder™
- CA SiteMinder®
- CA SiteMinder® for Secure Proxy Server
- CA Layer 7

Contents

Chapter 11: Single Sign-on Configuration 107

Chapter 1: Partnership Federation Introduction

This section contains the following topics:

Product and Configuration Overview

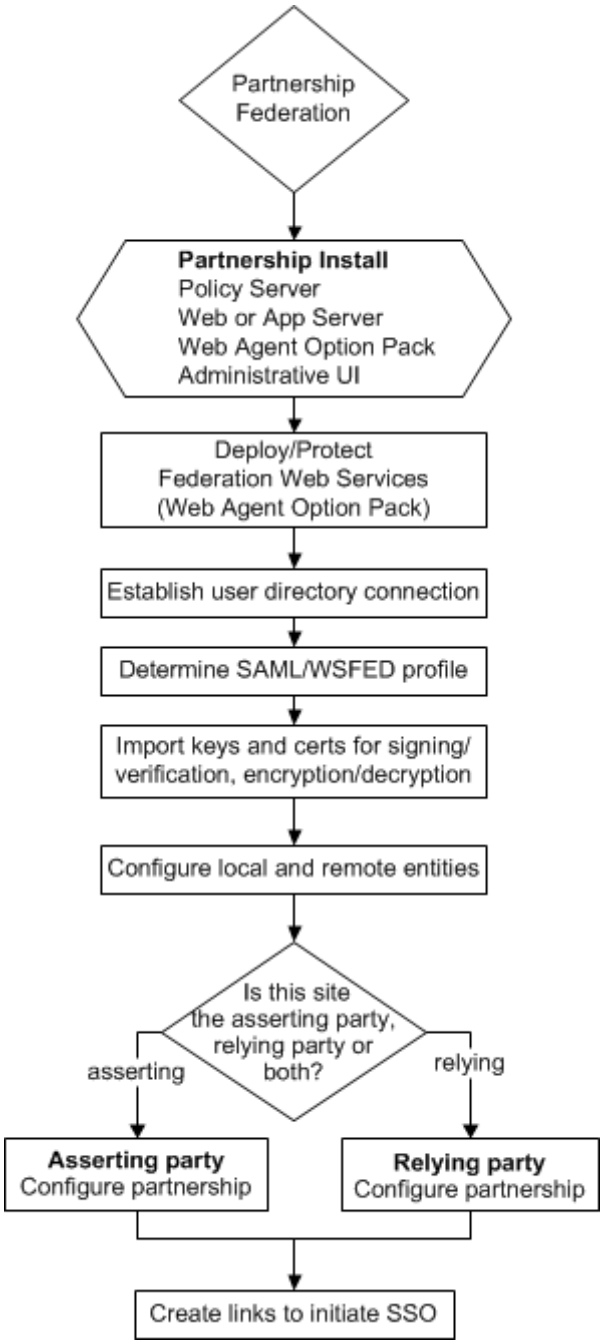
Federated partnerships enable identity information to be flexible and portable. Partnership federation offers secure single sign-on and single logout across a network of trusted business partners.

CA SiteMinder partnership federation lets customers establish federated partnerships in a flexible way, together with or independent of a web access management system. Partnership federation offers an easy-to-deploy solution for standards-based federation. Using partnership federation, an organization can act as the asserting party or the relying party. The asserting party provides user authentication and assertion of identity. The relying party consumes a user identity to allow access to web resources and services.

Partnership federation supports the following profiles:

- SAML 1.1
- SAML 2.0
- WS-Federation

The following flow chart highlights the general process for configuring partnership federation.



Programmerless Federation

Programmerless federation is an HTTP-based approach for allowing the secure authentication, user disambiguation, inspection, and modification of SAML assertions. The advantage of programmerless federation is that applications can accomplish these tasks without having to use a language-specific SDK or other bindings.

Programmerless federation relies on HTTP/HTTPS requests and responses. These requests and responses are accessible through URLs and HTML-based protocols using web services that are an implementation of Representational State Transfer (REST) system architecture.

Any application can issue HTTP requests, read HTTP responses, and can parse XML to take advantage of the programmerless functionality.

An essential part of programmerless federation is its ability to secure the exchange of data. To secure data, CA SiteMinder uses an open-format cookie. The open-format cookie is a well-defined cookie format that supports strong encryption algorithms. The encrypted cookie secures the response between CA SiteMinder and the local or remote applications. This cookie can be written in any programming language that supports the same encryption and decryption algorithms that are supported by the open-format cookie, such as Perl or Ruby.

The following partnership federation features implement programmerless federation:

Delegated Authentication

Delegated authentication lets CA SiteMinder use a third-party web access management (WAM) system to perform the authentication of any user who requests a protected federated resource. The third-party WAM performs the authentication and then sends the federated user identity to CA SiteMinder.

HTTP/HTTPS requests and responses facilitate communication for provisioning.

Provisioning at the Relying Party

Provisioning is the process of creating client accounts with the necessary account rights and access privileges for accessing data and applications. Partnership federation provisioning can establish a new account for a user, or can populate an existing user account with information sent in a SAML assertion.

Remote provisioning is one of the CA SiteMinder provisioning methods. Remote provisioning uses an independent provisioning application to establish a user record. To pass assertion data, CA SiteMinder creates an encrypted cookie containing the data. This cookie is sent to the remote provisioning application, which is responsible for creating the user account.

HTTP/HTTPS requests and responses facilitate communication for provisioning.

Intended Audience

This guide assumes that you understand the following concepts:

- Basic SAML ans WS-Federation fundamentals
- Federation bindings.
- Federated profiles, such as Single Sign-on (SSO), Single logout (SLO), and Single Sign Out
- Public Key Infrastructure (PKI) fundamentals
- Secure Socket Layer communication basics

Terminology Used in this Guide

In addition to standard federated SAML and WS-Federation binding and profile terminology, the following terms are used in this guide:

Partner Entity Terms

This guide uses the terms *asserting party* and *relying party* to identify sides of a federated relationship.

The party that generates assertions is referred to as the asserting party. The asserting party can be:

- SAML 1.x producer
- SAML 2.0 Identity Provider (IdP)
- WS-Federation Identity Provider (IP)

The party that consumes assertions for authentication purposes is referred to as the relying party. The relying party can be:

- SAML 1.x consumer
- SAML 2.0 Service Provider (SP)
- WS-Federation Resource Partner (RP)

A site can be act as an asserting party (producer/IdP/IP) and a relying party (consumer/SP/RP).

Open Format Cookie

A cookie that contains user identity information. The open-format cookie can be encrypted using FIPS or non-FIPS compatible algorithms, depending on how you generate it. You can create an open-format cookie using a CA SiteMinder® Federation SDK or you can create it manually using any programming language that supports UTF-8 encoding.

If you require a FIPS-encrypted open-format cookie, use an SDK to create the cookie and to read the cookie. The CA SiteMinder® Federation Java SDK can encrypt the cookie using a FIPS-compliant (AES) algorithm or a non-FIPS (PBE) algorithm. The CA SiteMinder® Federation .NET SDK can encrypt the cookie using only a FIPS-compatible algorithm.

Unified Expression Language

The Unified Expression Language (UEL) is a special Java expression syntax primarily for Java web applications. You can use the UEL for embedding expressions into web pages. For partnership federation, the UEL is the language you must use to define mappings between assertion attributes and application attributes at the relying party.

Navigating the Partnership Federation Dialogs

The Administrative UI provides configuration wizards to create and modify partnership federation objects. Follow the steps in the configuration wizard to navigate through the configuration steps for an object.

Chapter 2: Prerequisites for Partnership Federation

This section contains the following topics:

Prerequisites for a CA SiteMinder Asserting Partner

For CA SiteMinder to serve as the asserting partner, verify the following conditions:

- The Policy Server is installed.
- The Web Agent and the Web Agent Option Pack are installed. The Web Agent authenticates users and establishes a CA SiteMinder session. The Option Pack provides the Federation Web Services application. Be sure to deploy the FWS application on the appropriate system in your network.

  For more information, see the *Web Agent Option Pack Guid*e.
- Private keys and certificates are imported for functions that require signing and decrypting messages.
- SQL Query scheme and valid SQL queries are set up before selecting an ODBC database as a user directory for the partnership. This prerequisite is only necessary if you plan to use ODBC.
- A relying partner is set up within the federated network.

Prerequisites for a CA SiteMinder Relying Partner

For CA SiteMinder to serve as the relying partner, satisfy the following requirements:

- The Policy Server is installed.
- The Web Agent and the Web Agent Option Pack. The Web Agent authenticates users and establishes a CA SiteMinder session. The Option Pack provides the Federation Web Services application. Be sure to deploy the FWS application on the appropriate system in your network.

  For more information, see the *Web Agent Option Pack Guid*e.
- Private keys and certificates are imported for functions that require verification and encrypting of messages.

- An asserting partner is set up within the federated network.

Chapter 3: Getting Started with a Simple Partnership

This section contains the following topics:

Basic SAML 2.0 Partnership

One way to get started with partnership federation is by configuring a partnership. This chapter describes how to set up a basic SAML 2.0 federation partnership—single sign-on with SAML 2.0 POST profile. By starting with a basic configuration, you can complete the least number of steps to see how partnership federation works.

Note: This partnership focuses on SAML 2.0; however, the overall process is the same for SAML 1.1. The configuration settings at each step of the partnership can differ depending on the SAML protocol.

The chapter also describes the configuration of additional features, such as digital signing and single logout to reflect a real production environment. You can also add the Artifact binding to the configuration.

The sample network used in this chapter presupposes that CA SiteMinder is installed at both sites in the partnership. However, you can have CA SiteMinder at one site and a different SAML-compliant product at the other site and still engage in a partnership.

With CA SiteMinder at both sites, you have to understand the perspective from which you are configuring a partnership. To configure a complete partnership, you begin by defining a *partnership definition* at each site, one for each direction of communication from a given site. For example, if the local site is the Identity Provider (IdP), you configure the local IdP-to-remote SP partnership. This configuration is one partnership definition. To complete the partnership configuration, you configure the reciprocal local SP-to-remote IdP partnership at the local SP.

The partnership definition always distinguishes the local and remote entities. The local entity is the entity at the site from where you are configuring partnership federation. This environment is not necessarily the same as the one on which CA SiteMinder is installed, but the same domain. The remote entity is the entity at a partner that resides in a different domain from where you are configuring partnership federation.

The following process shows the steps for creating the basic partnership when CA SiteMinder is at both sites:

1. Establish a user directory connection.
2. Protect the authentication URL to establish a session.
3. Create the local and remote entities.
4. Configure the local IdP-to-SP partnership definition at the IdP.
5. Configure the local SP-to-IdP partnership definition at the SP.
6. Activate the partnership.
7. Test the partnership.

Sample Federation Network

The initial partnership that you are creating represents the following sample network. The URLs in the procedures and sample network are examples and do not resolve to any real site.

The Business Partners

- Identity Provider named IdP1
- Service Provider named SP1

SAML Profiles and Features

- SAML 2.0 with POST profile
- Single sign-on
- No signature processing
- FIPS_COMPAT mode

SSO Service URL at the IdP

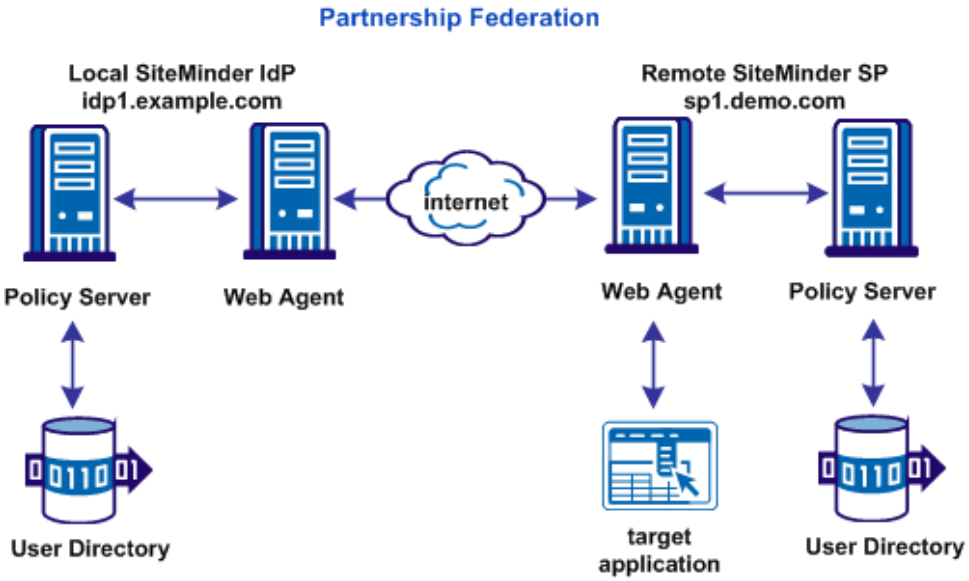
http://idp1.example.com:9090/affwebservices/public/saml2sso

Assertion Consumer Service URL at the SP

http://sp1.demo.com:9091/affwebservices/public/saml2assertionconsumer

Note: You need two systems with CA SiteMinder installed to implement this sample network.

The following figure shows the sample partnership with CA SiteMinder at both partners.



Confirm that Required Components are Installed

To use partnership federation, the following components are required:

- Policy Server
- Administrative UI
- Web Agent
- Web Agent Option Pack

  The Web Agent Option pack includes the Federation Web Services (FWS) application. FWS is a required component for federation.

  To install the Web Agent Option Pack and deploy FWS, see the *Web Agent Option Pack Guide*.

This simple partnership deployment example assumes that these components are installed and working.

Configure the IdP Partner

The configuration process that follows is from the perspective of an administrator at IdP1. Therefore, IdP1 is the local IdP.

The following process establishes the IdP partner:

1. Log in to the Administrative UI.
2. Establish a user directory connection.
3. Identify the IdP and SP entities.
4. Create a SAML2 IdP->SP partnership.
5. Follow the partnership wizard and configure the minimum required settings.

Establish a User Directory Connection at the IdP

Before you can establish a partnership, define a connection to a user directory. The IdP user directory consists of user records for which the Identity Provider generates assertions.

The following steps specify how to configure a user directory in the Administrative UI. The directory named IdP LDAP contains user1 and user2.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Infrastructure, Directory, User Directories.
3. Click Create User Directory.

   The User Directory dialog opens.
4. Complete the following fields:

   Name

   IdP LDAP

   NameSpace

   LDAP

   Server

   www.idp.demo:42088

5. Complete the following field in the LDAP Settings section:

   Root

   dc=idp,dc=demo

   Accept the defaults for the other values.

   Complete the following field in the LDAP User DN Lookup:

   Start

   uid=

   End

   ,ou=People,dc=idp,dc=demo

6. Click View Contents to verify you can view the contents of the directory.
7. Click Submit.

Protect the Authentication URL to Establish a Session

A user must have a session at the IdP Policy Server for the Policy Server to generate an assertion. To establish the session, protect an authentication URL with a policy so that the user is presented with an authentication challenge. The user then logs in and a session is established.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Infrastructure, Agents, Create Agent.

   Ceate a web agent named Agent1.

3. Select Policies, Domain, Domains, Create Domain.

   Create a policy domain for the authentication URL. Add the user directory that contains the users who get challenged.

4. Select the users that must have access to the resources that are part of the policy domain.
5. Select the Realms tab and define a realm for the policy domain with the following values:

   Agent

   Agent1

   Resource Filter

   /affwebservices/redirectjsp

Default Resource Protection

Protected

Authentication Scheme

Basic

Persistent Session

Select the Persistent check box in the Session section of the realm dialog for the HTTP-Artifact profile and to store session information. Session information is required for features such as single logout and for an attribute authority.

6. In the Rules section of the realm dialog, click Create Rule. Complete the fields with the following values:

Resource

/*

The asterisk means that the rule applies to all resources in the realm.

Allow/Deny and Enable/Disable

Allow Access

Enabled check box is selected.

Action

Web Agent actions

Get, Post, Put

7. Select the Policies tab and create a policy that includes the following components:
   - The set of users you selected in your user directory.
   - The realm that contains the redirectjsp application and the associated rule.

A policy now protects the authentication URL.

Configure the Partnership Entities

After you establish the user directory connection, identify both sides of the partnership. In the Administrative UI, each partner is referred to as an entity.

The following procedures tell you what values to provide for the local and remote entities. In a real network configuration, each side can create a local entity, export the local entity to a metadata file, then exchange files. Each side can then define the remote entity.

To create the local IdP

1. Select Federation, Partnership Federation, Entities.
2. Click Create Entity in the Federation Entity List.
3. Make the following selections in the first step of the entity wizard then click Next.

   Entity Location

   Local

   New Entity Type

   SAML2 IDP
4. Complete the following fields in the second step of the wizard then click Next.

   Entity ID

   idp1

   This value identifies the entity to the partner.

   Entity Name

   idp1

   This value identifies the entity object internally in the database. The partner is not aware of this value.

   Base URL

   http://idp1.example.com:9090

   Leave the other settings as they are.

   Note: The Entity Name can be the same value as the Entity ID. However, do not share the values with any other entity at the site.
5. Review the settings in the last step and click Finish.

You return to the Entities window.

To create the SP Entity

1. Begin at the Entities window.
2. Click Create Entity in the Federation Entity List.

   The Create Entity dialog displays.
3. Make the following selections in the first step of the entity wizard then click Next.

   Entity Location

   Remote

   New Entity Type

   SAML2 SP
4. Complete the fields in the second step of the wizard as follows, then click Next.

   Entity ID

   sp1

   This value identifies the entity to the partner.

   Entity Name

   sp1

   This value identifies the entity object internally in the database. The partner is not aware of this value.

   Assertion Consumer Service URLs

   Index

   0

   Binding

   HTTP-Post

   URL

   http://sp1.demo.com:9091/affwebservices/public/saml2assertionconsumer

   Default

   Select the check box for the entry.

   Leave the other settings as they are.
5. Review the settings in the last step and click Finish.

The remote SP entity is configured.

After the local and remote entity are configured, create a partnership.

Create the IdP-to-SP Partnership

After you create federation entities, follow the partnership wizard to configure the IdP ->SP partnership. The wizard begins with the basic partnership parameters.

Follow these steps:

1. Select Federation, Partnership Federation, Partnerships.
2. Click Create Partnership.
3. Select SAML2 IdP -> SP.

   Selecting this option indicates that you are the local IdP.

   You come to the first step in the partnership wizard.
4. Complete the fields with the following values:

   Partnership Name

   TestPartnership

   Local IDP ID

   idp1

   (selected from the pull-down list)

   Remote SP ID

   sp1

   (selected from the pull-down list)

   Base URL

   http://idp1.example.com:9090

   Skew Time (Seconds)

   Accept the default
5. Move the IDP LDAP directory from the Available Directories list to the Selected Directories list.
6. Click Next to go to the Federation User step.

Specify Federation Users for Assertion Generation

In the Federation Users dialog, select the users for which the IdP generates assertions.

Follow these steps:

1. Accept the defaults.
2. Click Next to continue.

By accepting the defaults, you indicate that CA SiteMinder can generate assertions for all users in the user directory.

Add a Name ID to the Assertion

The Assertion Configuration step lets you specify the format and value of the NameID and the attributes that identify a user. These attributes are included in the assertion.

Note: NameID is always included in the assertion.

In this configuration, specify only the Name ID. Do not add any other attributes.

Follow these steps:

1. From the Assertion Configuration step, enter values for the following fields:

   Name ID Format

   Unspecified

   Name ID Type

   Static

   Value

   GeorgeC
2. Click Next to move on and set up single sign-on (SSO).

Set Up Single Sign-on at the IdP

To establish single sign-on between partners, configure the SSO settings.

Follow these steps:

1. Begin at the SSO and SLO step in the partnership wizard.
2. In the Authentication section, specify the following entries:

   Authentication Mode

   Local

   Authentication URL

   http://webserver1.example.com/affwebservices/redirectjsp/redirect.jsp

   In this example, webserver1 identifies the web server with the Web Agent Option Pack. The redirect.jsp file is included with the Web Agent Option Pack installed at the Identity Provider site.

   Important! Protect the Authentication URL with an access control policy.

   Configure AuthnContext

   Accept the default

   Authentication Class

   Accept the default

3. In the SSO section, specify the following entries:

   SSO Binding

   HTTP-POST

   Assertion Consumer URL

   http://sp1.demo.com:9091/affwebservices/public/saml2assertionconsumer

4. Click Next to move to the Signature and Encryption step.

Disable Signature Processing

For the purposes of this simple partnership, disable signature processing. However, in a production environment, the Identity Provider must sign assertions.

Follow these steps:

1. From the Signature and Encryption step, select Disable Signature Processing.
2. Click Next to move to the next step.

Confirm the IdP-to-SP Partnership Settings

You have completed the partnership definition for one side of the federation partnership. Verify the settings.

Follow these steps:

1. In the Confirm dialog, review the settings for the partnership.
2. To modify a setting, click Modify in any of the sections.
3. Click Finish when you are satisfied with the configuration.

The IdP side of the partnership is complete. Define the SP side of the partnership on a different system than the IdP system.

Configure the SP Partner

The configuration process that follows is from the perspective of an administrator at the SP, in this example, SP1. Therefore, SP1 is the local SP.

The following process establishes the SP partner.

1. Log in to the Administrative UI.
2. Establish a user directory connection.
3. Identify the IdP and SP entities.
4. Create a SAML2 SP->IdP partnership.
5. Follow the partnership wizard and configure the minimum required settings.

Establish a User Directory Connection at the SP

The SP user directory consists of user records for which the Service Provider uses for authentication. The following steps specify how to configure a user directory in the Administrative UI. The directory named SP LDAP contains users user1 and user2.

To configure a user directory

1. Log in to the Administrative UI.
2. Select Infrastructure, Directory, User Directories.
3. Click Create User Directory.

   The User Directory dialog opens.

4. Complete the following field:

   Name

   SP LDAP

5. Complete the following fields in the Directory Setup section:

   Namespace

   LDAP

   Server

   www.sp.demo:32941

6. Complete the following fields in the LDAP Search section:

   Root

   dc=sp,dc=demo

   Accept the defaults for the other values.

7. Complete the following fields in the LDAP User DN Lookup section:

   Start

   uid=

   End

   ,ou=People,dc=sp,dc=demo

8. Click View Contents to verify that you can view the contents of the directory.
9. Click Submit.

Identify the Partnership Entities

After you establish the user directory connection, identify the local and remote sides of the partnership. In the Administrative UI, each partner is referred to as an entity.

The following procedures tell you what values to provide for the local and remote entities. Typically, each side creates a local entity, exports the local entity to a metadata file, and then exchanges the files. Each side can then define the remote entity.

To create the local SP

1. Select Federation, Partnership Federation, Entities.
2. Click Create Entity.

3. Make the following selections in the first step of the entity wizard then click Next.

   Entity Location

   Local

   New Entity Type

   SAML2 SP

4. Complete the fields in the second step as follows, then click Next.

   Entity ID

   sp1

   This value identifies the entity to the partner.

   Entity Name

   sp1

   This value identifies the entity object internally in the database. The partner is not aware of this value.

   Base URL

   http://sp1.demo.com:9091

   Note: The entity ID and name must be the same as you specified for the remote SP entity at the Identity Provider.

5. Review the settings and click Finish.

You return to the Entities window. Configure the remote partner.

To create the remote IdP

1. Begin at the Entities window.
2. Click Create Entity.
3. Make the following selections in the first step of the entity wizard then click Next.

   Entity Location

   Remote

   New Entity Type

   SAML2 IDP

4. Complete the fields in the second step of the wizard as follows:

   Entity ID

   idp1

   This value identifies the entity to the partner.

   Entity Name

   idp1

   This value identifies the entity object internally in the database. The partner is not aware of this value.

   Note: The entity ID and name must be the same as on the Identity Provider side.

   SSO Service URL Group Section

   Binding

   HTTP-Redirect

   URL

   http://idp1.example.com:9090/affwebservices/public/saml2sso

5. Review the settings and click Finish.

After the local entity and remote entity are configured, you can create a partnership.

Create the SP-to-IdP Partnership

After you have created the partnership entities, follow the partnership wizard to configure the SP-> IdP partnership.

Follow these steps:

1. Select Federation, Partnership Federation, Partnerships.
2. Click Create Partnership.
3. Select SAML2 SP->IdP.

   You come to the first step in the partnership wizard.

4. Complete the fields with the following values:

   Partnership Name

   DemoPartnership

   Local SP ID

   sp1

Remote IDP ID

idp1

Base URL

http://sp1.demo.com:9091

Skew Time (Seconds)

Accept the default

5. Move the SP LDAP directory from Available Directories to the Selected Directories.
6. Click Next to go to the User Identification step.

Specify the User Identification Attribute

Designate which attribute from the assertion identifies a user. CA SiteMinder uses the identity attribute value to locate the user record in the user directory at the SP.

To specify the user identification attribute

1. Go to the User Identification step.
2. In the Choose Identity Attribute from Assertion section, accept the default, Use Name ID.
3. In the Map Identity Attribute to User Directories section, specify the following entry:

   LDAP Search Specification

   uid=%s

   This entry instructs CA SiteMinder to replace the variable (%s) with the value of the Name ID attribute from the assertion. CA SiteMinder then matches the value with the Name column in the sample users database. If a match is found, the user is disambiguated and allowed to access the target resource.
4. In the Federated Users section, accept the defaults. All users in the user directory are considered federated users.
5. Click Next to configure single sign-on.

Configure Single Sign-on at the SP

To establish single sign-on between partners, configure the SSO settings.

Follow these steps:

1. Begin at the SSO and SLO step.
2. Select HTTP-POST for the SSO Profile.

3. Specify the following values in the Remote SSO Service URLs section:

   Binding

   HTTP-Redirect

   URL

   http://idp1.example.com:9090/affwebservices/public/saml2sso

4. Click Next until you reach the Signature and Encryption step.

   Skip the Configure AuthnContext step.

Disable Signature Processing

For the purposes of this simple partnership, disable signature processing. However, in a production environment, the Identity Provider must sign assertions.

Follow these steps:

1. From the Signature and Encryption step, select Disable Signature Processing.
2. Click Next to move to the next step.

Specify the Target at the SP

The Application Integration step is where you specify the target resource and how CA SiteMinder redirects the user to the target resource.

Follow these steps:

1. Select No Data for the Redirect Mode field.
2. Specify the target resource at the SP in the Target field.

   In this sample partnership, this target is:

   http://spapp.demo.com:80/spsample/welcome.html

3. Ignore the remaining sections of the dialog.
4. Click Next to move to the Confirm step.

Confirm the SP Partner Settings

You have completed the partnership for the local SP side of the federation partnership.

Follow these steps:

1. In the Confirm dialog, review the settings for the SP partner.
2. To modify a setting, click Modify in the appropriate section.
3. Click Finish when you are satisfied with the configuration.

The SP side of the partnership is now configured.

Activate the Partnership

Each side of the partnership is defined, so you can now activate the partnership.

CA SiteMinder is installed at both sites in the partnership so you must activate the partnership at the IdP and SP.

To activate a partnership

1. Select Federation, Partnership Federation, Partnerships.
2. Find the entry in the Federation Partnership List that you want to activate. Verify that the value in the Status column is Defined. If the status is Incomplete, edit the partnership. Confirm all the required settings are configured.
3. Select Action, Activate next to the partnership entry that you want to activate.

   The Confirm Activate dialog displays.
4. Click Yes.

   The partnership is activated, and the value in the Status column is Active.

Test the Partnership (POST Profile)

After the partnership is configured, test single sign-on between the two partners.

Testing involves:

- Creating a web page to initiate single sign-on.
- Creating a target web page that serves as the requested federated resource.
- Testing single sign-on.

After you test the basic partnership, you can make more changes to the sample configuration.

Create a Web Page to Initiate Single Sign-on

For testing purposes, create your own html page with a link that initiates single sign-on. You can initiate single sign-on from the IdP or SP. This example illustrates SP-initiated single sign-on.

Follow these steps:

1. Create the sample HTML page at the SP site. Include a hard-coded link to the AuthnRequest service at the SP, as follows:

   Link to Test POST Single Sign-on

   This link instructs the AuthnRequest Service to redirect the user to the specified Identity Provider to retrieve the authentication context.

2. Save the web page under the name testsso.html.
3. Copy testsso.html to the web server document root directory, under a subfolder named /spsample.

   For this sample network, the target web server is http://spapp.demo:80.

Create a Target Resource

The last step that is required to test single sign-on is to create a target resource.

Follow these steps:

1. Create the sample HTML page at the SP site and include a message, such as:

   Welcome to SP1

   Single Sign-on is successful

2. Save the web page under the name welcome.html.
3. Copy welcome.html to the web server document root directory, under the subfolder /spsample.

   For this sample network, the target web server is http://spapp.demo.com:80.

Test POST Single Sign-on

After you set up the sample web pages, test single sign-on and verify that that partnership configuration is successful.

Follow these steps:

1. Verify that both sides of the partnership are activated in the Administrative UI.
2. Open up a browser.
3. Enter the URL for the web page that includes the link to trigger single sign-on. For this example, enter the following URL:

   http://spapp.demo.com:80/spsample/testsso.html

   After you have entered the URL, a page is displayed with a link that reads Link to Test POST Single Sign-on.
4. Click **Link to Test POST Single Sign-on**.

   Single sign-on is initiated. The user is redirected from the Service Provider to the Identity Provider.

After the Identity Provider establishes a session, it directs the user back to the target resource at the Service Provider, which is welcome.html. You see the sample welcome page that you created at the SP. The displayed page indicates single sign-on was successful.

Enable Signature Processing

Digitally signing assertions is required in a SAML 2.0 POST single sign-on. For signing and verification tasks, CA SiteMinder uses a private key/certificate pair.

Before any transaction or runtime actions, an administrator at IdP1 sends a file to SP1 that contains a certificate (public key). This key is associated with the private key. IdP1 uses the public key to sign assertions. An administrator at SP1 adds the certificate to its certificate data store.

When the single sign-on transaction occurs, IdP1 signs the assertion with its private key. SP1 receives the assertion and verifies the assertion signature using the certificate in its certificate data store.

Configure Signature Processing at the IdP

For HTTP-POST single sign-on, Idp1 is required to sign assertions. The IdP has to sign the assertion using a private key stored in the certificate data store.

Note: The example assumes that you have a file from which you can import a key/certificate pair. Alternatively, a private key/certificate pair is already in the certificate data store.

To configure signing

1. Select Federation, Partnership Federation, Partnerships.
2. Select Action, Deactivate next to the entry for TestPartnership, which is the IdP ->SP partnership.

   Deactivation is required before editing.
3. Click Action, Modify next to the TestPartnership entry.

   The partnership wizard opens.
4. Select the Signature and Encryption step.
5. In the Signature section, complete the following tasks:
   a. Clear Disable Signature Processing.
   b. Click Import next to the Signing Private Key Alias field.

      The Import Certificate/Private Key window opens.
6. Complete the import wizard as follows:
   a. Select the file from where you are importing the private key/certificate pair.
   b. For a pkcs#12 file, supply the password that encrypts the file. You already have this password.
   c. Select the certificate entry from the file that you want to import and enter a value for the Alias, such as cert1.
   d. Confirm the selection and click Finish.

   You return to the Federation Partnerships list.
7. Select Action, Modify for the partnership entry.
8. Go to the Signature and Encryption step. In the dialog, notice that the key/certificate that you imported is now available from the Signing Private Key Alias drop-down list.
9. Select the alias, cert1 and click Next.

10. Review the settings in the Confirm dialog and click Finish.

    You return to the Partnerships window.

11. Reactivate the partnership by selecting Action, Activate next to the TestPartnership entry.

Signature processing is now configured at the IdP.

Configure Signature Processing at the SP

SP1 is required to verify the signature of an assertion. Before a transaction, SP1 has received the certificate (public key) from IdP1. This certificate is for the private key IdP1 used to sign the assertion. This certificate is imported into the SP1 certificate data store.

To configure signature verification

1. Select Federation, Partnership Federation, Partnerships.

   The Partnerships window opens.

2. Select Action, Deactivate next to the entry for DemoPartnership.

   Deactivation is required before editing.

3. Click Action, Modify next to the DemoPartnership entry.

   The partnership wizard opens.

4. Select the Signature and Encryption step.

5. In the Signature section, complete the following tasks:

   a. Clear Disable Signature Processing.

   b. Click Import next to the Verification Certificate Alias field.

      The Import Certificate/Private Key window opens.

6. Complete the import wizard as follows:

   a. Select the file from where you are importing the certificate.

   b. Select the certificate entry from the file that you want to import and enter a value for the Alias, such as cert1.

   c. Confirm the selection and click Finish.

   You return to the Federation Partnership List.

7. Select Action, Modify for the partnership entry.

8. Go to the Signature and Encryption step. In the dialog. Notice that the key/certificate that you imported is now available from the Signing Private Key Alias drop-down list.

9. Select the alias, cert1, for the certificate and click Next.

10. Review the settings in the Confirm dialog and click Finish.

    You return to the Partnerships window.

11. Reactivate the partnership by selecting Action, Activate next to the DemoPartnership entry.

Signature verification is now configured at the SP.

Add Single Logout

The single logout protocol (SLO) results in the simultaneous end of all user sessions for the browser that initiated the logout. Configuring single logout helps ensure that no sessions are left open for unauthorized users to gain access to resources at the Service Provider.

Important! To see the SLO settings, enable the session store using the Policy Server Management Console. For instructions about using the Management Console, see the *Policy Server Administration Guide* for instructions.

Configure Single Logout at the IdP

Configure single logout at Idp1.

Follow these steps:

1. Select Federation, Partnership Federation, Partnerships.

   The Partnerships windows displays.

2. Select Action, Deactivate next to the TestPartnership entry.

   Deactivate a partnership before editing it.

3. Click Action, Modify next to the TestPartnership entry.

   The partnership wizard opens.

4. Select the SSO and SLO step.

5. In the SLO section, configure the following fields:

   SLO Binding

   HTTP-Redirect

   SLO Confirm URL

   http://idp1.example.com:9090/idpsample/SLOConfirm.html

   This link is the confirmation page at the site that initiated single logout, in this case, IdP1. If single logout completes successfully, the user is redirected to this page.

6. Click Add Row in the SLO Service URLs table and complete the following field:

   SLO Location URL

   http://sp1.demo.com:9091/affwebservices/public/saml2slo

   This link indicates that the single logout request is sent to the remote SP.

7. Select the row that you configured in the Select column.
8. Click the Confirm step in the wizard and review the configuration.
9. Click Finish.

   You return to the Partnerships window.

10. Reactivate the partnership by selecting Action, Activate next to the TestPartnership.

Single logout is now added to the configuration at IdP1.

Configure Single Logout at the SP

Configure single logout at SP1.

To configure single logout at the SP

1. Select Federation, Partnership Federation, Partnerships.

   The Partnerships window displays.

2. Select Action, Deactivate next to the entry for Demo Partnership.

   Deactivate a partnership before editing it.

3. Click Action, Modify next to the entry for DemoPartnership.

   The dialog for the first step of the Partnership wizard opens.

4. Click the SSO and SLO step.
5. In the SLO section, configure the following fields:

   SLO Binding

   HTTP-Redirect

   SLO Confirm URL

   http://sp1.demo.com:9091/spsample/SLOConfirm.html

   This URL is the single logout confirmation page at the site that initiated the logout.

6. Click Add Row in the SLO Service URLs table and complete the following field:

   SLO Location URL

   http://idp1.example.com:9090/affwebservices/public/saml2slo

   This URL is where the single logout request is sent.

7. Select the row that you configured in the Select column.
8. Click the Confirm step in the wizard and review the configuration.
9. Click Finish.

   You return to the Partnerships window.

10. Reactivate the partnership by selecting Action, Activate next to the DemoPartnership entry in the Federation Partnership List.

Single logout is now configured at the SP.

Test Single Logout

After you configure single logout, test it. For this test, single logout is initiated at SP1.

Initiating single logout from the SP requires that you have two web pages to initiate and confirm single logout.

- Using welcome.html, add a link to this page that directs the browser to the Single Logout Service at IdP1. This link has the following syntax:

  Log Me Out

- Create a confirmation page named SLOConfirm.html with a logout confirmation message, such as:

  You have successfully logged out

Copy both these pages to your web server root directory under the subfolder /spsample.

Note: Complete an SSO transaction so you can test SLO.

Follow these steps:

1. Verify that both sides of the partnership are activated in the Administrative UI.
2. Configure and test single sign-on according to the previously documented instructions.

   If single sign-on is successful, the welcome page is displayed in the browser.
3. Keep the browser open and click the link **Log Me Out** on the welcome page.

   If successful, you are redirected to the confirmation page that displays the message:

   You have successfully logged out.

Set Up the Artifact Profile for SSO

The basic partnership began with HTTP-POST binding for single sign-on. However, your partnership can use the SAML 2.0 Artifact profile.

The configuration for the HTTP-Artifact binding is the same as the configuration for POST binding, until the SSO and SLO steps in the wizard.

Configure Artifact SSO at the IdP

This procedure shows you how to configure the HTTP-Artifact profile for SSO.

Follow these steps:

1. From the Administrative UI, Select Federation, Partnership Federation, Partnerships.

   The Partnerships window displays.
2. Select Action, Deactivate next to the entry for TestPartnership.

   Deactivation is required before editing.
3. Click Action, Modify next to the entry for TestPartnership.

   The partnership wizard opens.
4. Click the SSO and SLO step.
5. Keep the existing settings in the Authentication section.

6. In the SSO section, specify the following entries:

   SSO Binding

   HTTP-Artifact

   Artifact Protection Type

   Partnership

   Leave the remaining settings as is.

7. Add a row to the Assertion Consumer Service URLs table and use the following settings:

   Binding

   HTTP-Artifact

   URL

   http://sp1.demo.com:9091/affwebservices/public/saml2assertionconsumer

   This URL is the same one used for the POST profile.

8. In the Back Channel section, select the following authentication method for the Incoming Configuration:

   Authentication Method

   No Auth

9. Skip the other sections in the dialog.
10. Go to the Confirm step and review the configuration.
11. Click Finish to complete the configuration.

Artifact binding is now configured at Idp1.

Configure Artifact SSO at the SP

This procedure shows you how to configure the HTTP-Artifact profile for SSO.

Follow these steps:

1. Select Federation, Partnership Federation, Partnerships.

   The Partnerships window displays.

2. Select Action, Deactivate next to the entry for Demo Partnership.

   Deactivation is required before editing.

3. Click Action, Modify next to the DemoPartnership entry.

   The partnership wizard opens.

4. Click the SSO and SLO step.

5. In the SSO section, specify the following entries:

   SSO Profile

   HTTP-Artifact

   SSO Service URL

   Keep the same URL that was configured for HTTP-POST single sign-on.

6. Click Add Row in the Remote SOAP Artifact Resolution URLs table. Enter the following settings:

   Index

   1

   URL

   http://idp1.example.com:9090/affwebservices/public/saml2ars

7. Select this entry in the Select column of the table.

8. In the Back Channel section, select the following authentication method for the Outgoing Configuration:

   Authentication Method

   No Auth

9. Click Next until you reach the Application Integration step.

Specify the Target at the SP

The Application Integration step is where you specify the target resource and how CA SiteMinder redirects the user to the target resource.

Follow these steps:

1. Select No Data for the Redirect Mode field.
2. Specify the target resource at the SP in the Target field.

   In this sample partnership, this target is:

   http://spapp.demo.com:80/spsample/welcome.html
3. Ignore the remaining sections of the dialog.
4. Click Next to move to the Confirm step.

Test the Partnership (Artifact SSO)

When each side of the partnership is operating, test single sign-on between the two partners.

When IdP1 receives the request, it generates the artifact. The artifact is then sent to the SP1.

After SP1 receives the artifact, it redirects the request back to IdP1. The IdP retrieves the assertion and returns it to SP1.

Create a Web page to Initiate Single Sign-on (Artifact)

For testing purposes, create your own html page with a link that initiates single sign-on. You can initiate single sign-on from the IdP or SP. This example illustrates SP-initiated single sign-on.

Follow these steps:

1. Create the sample HTML page at the SP site and include a hard-coded link to the AuthnRequest service at the SP, as follows:

   <a href="http://sp1.demo.com:9091/affwebservices/public/
   saml2authnrequest?ProviderID=idp1.example.com:9090&
   ProtocolBinding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact>
   Link for ARTIFACT Single Sign-on

   This link instructs the AuthnRequest Service to redirect the user to the specified Identity Provider to retrieve the user authentication context.

2. Save the web page under the name testartifact.html.
3. Copy testartifact.html to the web server document root directory, under the subfolder /spsample.

   For this sample network, the target web server is http://spapp.demo:80.

Create a Target Resource

The last step that is required to test single sign-on is to create a target resource.

Follow these steps:

1. Create the sample HTML page at the SP site and include a message, such as:

   Welcome to SP1

   Single Sign-on is successful

2. Save the web page under the name welcome.html.
3. Copy welcome.html to the web server document root directory, under the subfolder /spsample.

   For this sample network, the target web server is http://spapp.demo.com:80.

Test Artifact Single Sign-on

After you have set up the sample web pages, test single sign-on and verify that the partnership configuration is successful.

Follow these steps:

1. Verify that both sides of the partnership are activated.
2. Open up a browser.
3. Enter the URL for the web page that triggers single sign-on, as follows:

   http://spapp.demo.com:80/spsample/testartifact.html

   Note: The target web server is a different server than the one where CA SiteMinder resides.

   When entering the URL, a page is displayed with a link that reads Link to Test ARTIFACT Single Sign-on.
4. Click **Link to Test ARTIFACT Single Sign-on** and single sign-on is initiated.

   The user is redirected from the SP to the Identity Provider.

After the Identity Provider establishes a session, it directs the user back to the target resource at the Service Provider, which is welcome.html. You see the sample welcome page that you created at the SP. The displayed page lets you know single sign-on was successful.

Configuration Procedures Beyond the Simple Partnership

The simple partnership provides an overview of configuring federated partnerships using partnership federation.

The remaining chapters in the guide provide detailed procedures for every task you can perform. For detailed configuration instructions, use these procedures as well as the Help in the Administrative UI.

More information:

Federation Entity Configuration (see page 61)
Partnership Creation and Activation (see page 71)

Chapter 4: Federation Features Requiring the Session Store

The session store holds data for the following federation features:

- HTTP-Artifact single sign-on (SAML 1.x or 2.x)

  A SAML assertion and the associated artifact are generated at the asserting party. The artifact identifies the generated assertion. The asserting party returns the artifact to the relying party. The relying party uses the artifact to retrieve the assertion, which the asserting party stores in the session store.

  A persistent session is required for this process to work.

  Note: The SAML POST profile does not store assertions in the session store.

- HTTP-POST single use policy (SAML 2.0 and WS-Federation)

  The single use policy feature prevents assertions from being reused at the relying party to establish a second session. The relying party stores time-based data about the assertion, which is known as expiry data, in its session store. Expiry data verifies that the assertion is only used one time.

  A session store is required at the relying party, but a persistent session is not required.

- Single logout (SAML 2.0)

  If single logout is enabled, either partner can stores information about the user session. The session information is kept in the session store. When a single logout request is completed, the session information for the user is removed, invalidating the session.

  A persistent session is required at the Identity Provider and Service Provider.

- Sign-out (WS-Federation)

  If sign-out is enabled, user context information is placed in the session store. This information enables the Policy Server to generate a sign-out request. When a sign-out request is completed, the session information for the user is removed, invalidating the user session.

  A persistent session is required at the Identity Provider and Resource Partner.

- Authentication Session Variables Persistence (all profiles)

  You can select the option Persist Authentication Session Variables when configuring federation at a relying party. This option instructs the Policy Server to save authentication context data in the session store as session variables. The Policy Server has access to these variables for use in authentication decisions.

- Assertion Attributes Persistence (all profiles)

  You can select Persist Attributes as a redirect mode at the relying party. The redirect mode determines how a user is redirected to the target application. This mode instructs the Policy Server to store assertion attributes in the session store so they can be supplied as HTTP header variables.

- Authentication Request POST Binding (SAML 2.0)

  For the IdP to handle an authentication request that is delivered using HTTP-POST binding, the IdP must store the request in the session store.

Enable the session store to hold this type of user session, assertion, and expiry data.

Enable the Session Store

Enable the session store to hold data when using SAML artifact for single sign-on, single logout, and enabling the single use of a policy.

Enable the session store from the Policy Server Management Console.

Follow these steps:

1. Log in to the Policy Server Management Console.
2. Select the Data tab.
3. Select Session Store from the drop-down list in the Database field.
4. Select an available storage type from the drop-down list in the Storage field.
5. Select the Session Store enabled check box.

   If you are going to use persistent sessions in one or more realms, enable the Session Server. When enabled, the Session Server impacts Policy Server performance.

   Note: The Use Policy Store database option is disabled. For performance reasons, the session server cannot be run on the same database as the policy store.
6. Specify Data Source Information appropriate for the chosen storage type.
7. Click OK to save the settings and exit the Console.
8. Stop and restart the Policy Server.

Environments that Require a Shared Session Store

The following features require a shared session store to store SAML assertions and user session information.

To implement these features across a clustered Policy Server environment, set up the environment as follows:

- Configure the login realm for persistent sessions for all features *except* for an HTTP-POST single use policy.

  Persistent sessions are part of the realm configuration.

- For HTTP-Artifact single sign-on, share the session store at the Producer/Identity Provider site across all Policy Servers in the cluster.

  Sharing the session store verifies that all Policy Servers have access to assertions when each one receives a request for an assertion.

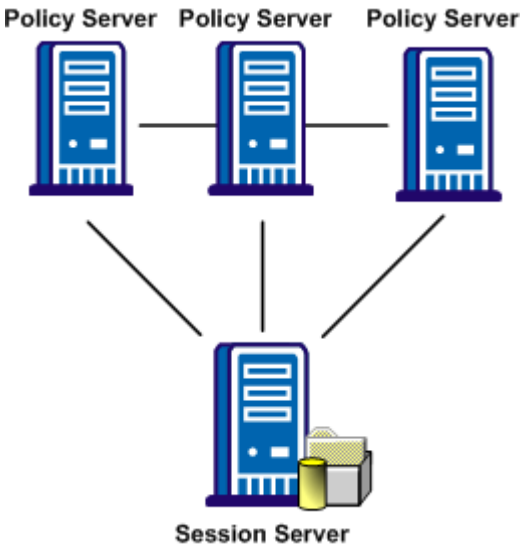
- For SAML 2.0 single logout and WS-Federation sign-out, share the session store at the asserting and relying party across all Policy Servers in the cluster.

  Sharing the session store verifies that all Policy Servers have access to user session data when each one receives a request for a session logout.

- For the HTTP-POST and WS-Federation single use policy feature, share the session store at the relying party across all Policy Servers in the cluster.

All Policy Servers that generate or consume assertions or process a persistent SMSESSION cookie must be able to contact the common session store. For example, a user logs in to example.com and gets a persistent session cookie for that domain. Every Policy Server that is handling requests for example.com must be able to verify that the session is still valid.

The following illustration shows a Policy Server cluster communicating with one session store:



To share a session store, use one of the following methods:

- Point all Policy Servers to one session store

  In the Policy Server Management Console, configure the Policy Server to use the designated session store.

- Replicate the session store across many session stores.

  For instructions on replicating a database, use the documentation for your database.

Chapter 5: User Directory Connections for Partnership Federation

Partnership federation looks up entries in a user directory to verify identities and retrieve user attributes for a given principal. At the asserting party, the federation partner generates assertions for the appropriate users, and authenticates each user against a user directory. At the relying party, the federation partner extracts the necessary information from an assertion and looks in the user directory for the appropriate user record.

Configure connections to existing user directories by selecting Infrastructure, Directory, User Directories in the Administrative UI. You are only establishing a connection to an existing user directory. You are not configuring a new user directory.

Note: To use an ODBC database in your federated configuration, set up the SQL query scheme and valid SQL queries before selecting an ODBC database as a user directory.

Configure connections to more than one directory if necessary. The directories do not have to be the same type.

For detailed information about user directories, see the *Policy Server Configuration Guide.*

Chapter 6: Require a CA SiteMinder Session by Protecting the Authentication URL

A user must have a session at the IdP Policy Server for the Policy Server to generate an assertion. To establish the session, the single sign-on service at the IdP redirects the user to an application by way of an authentication URL. Protect the authentication URL with a policy so that the user is presented with an authentication challenge. The user then logs in and a session is established.

The Authentication URL must point to the redirect.jsp file. For example:

http://webserver1.example.com/affwebservices/redirectjsp/redirect.jsp

In this example, webserver1 identifies the web server with the Web Agent Option Pack. The redirect.jsp file is included with the Web Agent Option Pack, which is installed at the Identity Provider.

After successful authentication, the redirect.jsp application redirects the user back to the single sign-on service for assertion generation.

Two steps are required to enable session creation:

1. Create the policy for the redirect.jsp file. (see page 57)
2. Specify the Authentication URL in a partnership (see page 59).

Create the Policy for the Redirect.jsp

A policy must protect the authentication URL to trigger the authentication challenge.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Infrastructure, Agents, Create Agent.

   To bind to the realm defined for the asserting party web server, create a web agent. Assign unique agent names for the web server.
3. Select Policies, Domain, Domains, Create Domain.

   Create a policy domain for the authentication URL. Add the user directory that contains the users who get challenged.
4. Select the users that must have access to the resources that are part of the policy domain.

5. Select the Realms tab and define a realm for the policy domain with the following values:

   Agent

   Agent for the asserting party web server. You created this agent in step 2.

   Resource Filter

   /affwebservices/redirectjsp

   This resource filter applies for a Web Agent and an SPS federation gateway.

   Default Resource Protection

   Protected

   Authentication Scheme

   Basic

   Persistent Session

   Select the Persistent check box in the Session section of the realm dialog for the HTTP-Artifact profile and to store session information. Session information is required for features such as single logout and for an attribute authority.

6. In the Rules section of the realm dialog, click Create Rule. Complete the fields with the following values:

   Resource

   /*

   The asterisk means that the rule applies to all resources in the realm.

   Allow/Deny and Enable/Disable

   Allow Access

   Enabled check box is selected.

   Action

   Web Agent actions

   Get, Post, Put

7. Select the Policies tab and create a policy that includes the following components:
   - The set of users you selected your user directory.
   - The realm that contains the redirectjsp application and the associated rule.

A policy now protects the authentication URL. An authentication challenge is triggered when the user is redirected to this URL. Finally, a session is created.

Specify the Authentication URL in a Partnership

After you configure a policy to protect the Authentication URL, specify this URL in the asserting-to-relying party partnership, such as the IdP->SP partnership.

The Authentication URL is set as part of the single sign-on configurations. In the Authentication section of the dialog, select **Local** for the Authentication Mode field and enter the complete Authentication URL. For example:

http://webserver1.example.com/affwebservices/redirectjsp/redirect.jsp

In this example, webserver1.example.com identifies the web server with the Web Agent Option Pack.

Chapter 7: Federation Entity Configuration

This section contains the following topics:

Methods to Create an Entity

Each partner in a federation partnership is considered a *federation entity*. Before you establish a partnership, define a local entity that represents the local partner and a remote entity that represents the remote partner.

The two ways to configure a federation entity are:

- Create an entity without using metadata (see page 61).
- Create an entity by importing metadata (see page 66).

Create an Entity without Using Metadata

Create an entity without metadata by using the following process:

1. Indicate an entity type.
2. Configure the specifics about that entity type.
3. Confirm the entity configuration.

Entity Type Choice

The first step in configuring an entity is to establish the entity type and determine the entity role.

To establish the entity type

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Entities.
3. Click Create Entity.

   The Create Entity dialog displays.

   Note: Click Help for a description of fields, controls, and their respective requirements.

4. Select *one* of the following options:

   Local

   Indicates that you are creating an entity that is local to your site.

   Remote

   Indicates that you are configuring an entity that represents the partner at the remote site.

5. Configure the remaining fields:

   New Entity Type

   Select the asserting or relying party.

   SAMLToken Type (WS-FED only)

   Select the token type, which defines the SAML format for the encrypted token that contains user credential information. Choose the Legacy option only if you want the token to comply with the SAML token type for WS-Federation 1.0.

6. Click Next to configure specifics about the entity.

Detailed Local Entity Configuration

After you have specified the entity type, configure the details of the entity. For a local entity, define the following information:

- Identification information about the entity
- Signature and encryption options
- Name ID formats and attributes

Follow these steps:

1. Begin at the Configure Entity step.
2. Complete any required fields for features and services for the local entity type you are configuring.

   Click Help for a description of the fields.

3. Click Next.

   The Confirm dialog is displayed.

Be aware of the following features:

Entity ID and Entity Name Settings

If the Entity ID represents a remote partner, the value must be unique. If the Entity ID represents a local partner, it can be reused on the same system.

The Entity Name identifies an entity object in the policy store. The Entity Name must be a unique value. This value is for internal use only; the remote partner is not aware of this value.

Note: The Entity Name can be the same value as the Entity ID, but do not share the value with other entities at the same site.

Signing and Encryption Features

For signing and encryption features, you must have the appropriate key/certificate entries in the certificate data store. If you do not have the appropriate key/certificate entries, click Import to import a private key/certificate pair from a file on your local system. You can also import trusted certificates.

Note: If you are using SAML 2.0 POST profile, signing assertions is required.

WSFED Attributes (WS-Federation only)

You can specify various service URLs and IDs for WS-Federation entites to communicate.

Name ID Formats

You can indicate the identifier types that the federated entity supports.

Assertion Attribute Configuration (asserting partners only)

You can configure the asserting party to include specific assertion attributes when it generates an assertion. The recommended method is to define these attributes at the entity level. The entity serves as a template for the partnership so any assertion attributes you define for the entity get propagated to the partnership. The benefit of defining assertion attributes at the entity is that it enables you to use an entity in more than one partnership.

If you want to add or remove assertion attributes for the partnership, make such modifications at the partnership level, not at the entity level.

Detailed Remote Entity Configuration

After you have specified the entity type, configure the details of the entity. For a remote entity type, define the following information:

- Identification information about the entity
- Signature and encryption options
- NameID and attribute information

Follow these steps:

1. Begin at the Configure Entity step.
2. Specify the Assertion Consumer Service URL. Examples:
   - If the SP is a site such as Google, the URL can be similar to:

     https://www.google.com/a/example.com/acs
   - If the SP is a site such as Salesforce.com, the URL can be similar to:

     https://login.salesforce.com/?saml=EK05LGnm40H7
   - If the SP is another business partner, the URL can be similar to:

     http://myserver.forwardinc.com:9080/samlsp/acs
3. Complete any other required fields for features and services for the remote entity type.

   Click Help for the field descriptions.
4. Click Next.

   The Confirm dialog is displayed.

Be aware of the following features:

Entity ID and Entity Name Settings

If the Entity ID represents a remote partner, the value must be unique. If the Entity ID represents a local partner, it can be reused on the same system.

The Entity Name identifies an entity object in the policy store. The Entity Name must be a unique value. This value is for internal use only; the remote partner is not aware of this value.

Note: The Entity Name can be the same value as the Entity ID, but do not share the value with other entities at the same site.

Signing and Encryption Features

For signing and encryption features, you must have the appropriate key/certificate entries in the certificate data store. If you do not have the appropriate key/certificate entries, click Import to import a private key/certificate pair from a file on your local system. You can also import trusted certificates.

Note: If you are using SAML 2.0 POST profile, signing assertions is required.

WSFED Attributes (WS-Federation only)

You can specify various service URLs and IDs for WS-Federation entites to communicate.

Name ID Formats

You can indicate the identifier types that the federated entity supports.

Assertion Attribute Configuration (asserting partners only)

You can configure the asserting party to include specific assertion attributes when it generates an assertion. The recommended method is to define these attributes at the entity level. The entity serves as a template for the partnership so any assertion attributes you define for the entity get propagated to the partnership. The benefit of defining assertion attributes at the entity is that it enables you to use an entity in more than one partnership.

If you want to add or remove assertion attributes for the partnership, make such modifications at the partnership level, not at the entity level.

Confirm the Entity Configuration

Review the entity configuration before saving it.

Follow these steps:

1. Review the settings in the entity dialog.
2. Click Back to modify any settings from this dialog.
3. Click Finish when you are satisfied with the configuration.

A new entity is configured.

Entity Configuration Changes from a Partnership

You can change an entity ID value for the remote entity from within the context of a single partnership configuration. However, changing the entity ID at the partnership level does not link the partnership to another entity, nor does it update the original entity. Modifications to an entity are a one-way propagation from the entity to the partnership. A change to the entity ID at the partnership level does not get propagated to the original entity.

Note: The entity ID you specify has to match what your remote partner is using.

Regard entity configurations as templates. Partnerships are created based on the entity templates so changing the partnership does not change the original entity template.

Refer to editing an entity from a partnership (see page 73) for more details about entities within a partnership.

Create an Entity by Importing Metadata

You can import data from a metadata file to create a federation entity. Importing the metadata reduces the amount of configuration for creating a partnership.

You can use metadata in the following ways:

- Import data from a remote partner to create a new remote entity.
- Import data from a remote partner to update an existing remote entity.
- Import data from a local entity to create a new local entity.

  This option can be useful to facilitate a migration from another federation product.

  Note: Federation does not support metadata imports to update or restore an existing partnership and local entity. To update an existing local entity, edit the entity and modify the settings that you want to change. You can import metadata only to create a *new* local entity.

The process for creating a metadata-based entity is as follows:

1. Select a metadata file for configuring a new entity.
2. Select an entity entry from the metadata file. The file can include several entities, but one entity per file is recommended.
3. (Optional) Select the certificates to import into the certificate data store. The certificates must be in the metadata file.

   These certificates can be used for authentication request verification, single logout response verification (SAML 2.0), and encryption (SAML 2.0).
4. Confirm the entity configuration.

Details about these steps are described in the next sections.

Metadata File Selection

The first step to create an entity from metadata is to select the metadata file.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Entities.
3. Click Import Metadata.

   The Import Metadata dialog opens.

   Click Help for the field descriptions.
4. Browse for the metadata file you want to use to create the entity.

5. Select whether to create a new local or remote entity, or update an existing remote entity.

   Note: The Policy Server does not support metadata imports to update an existing partnership and local entity. You can only create a new local entity. To update an existing local entity, edit the entity and modify the settings that you want to change. You can update the existing remote entities or you can create new remote entities.

6. Click Next to select entities from the file.

If you select a metadata file with expired entries, the next dialog that the UI displays contains a section listing the expired entries. You cannot select these expired entries; they are displayed for your reference. If all entities in a metadata file are expired, no entities are displayed. In this case, upload a new document.

Select an Entity to Import

This procedure assumes that you have already selected a metadata file to create an entity. Select the entity from the file.

Follow these steps:

1. Specify a name for the new entity in the Select Entity Defined in File dialog.

   If you are doing a local import to create an entity, define the partnership name.

2. Click on the option button to select the entity.

3. Click Next.

   The Import Certificates dialog displays if importing metadata for a remote entity and the document includes certificate data.

   If the metadata file that you imported contains certificate entries, you can import these entries.

Certificate Imports

To verify signed assertions, import certificates if the metadata includes them. If the metadata does not include certificates, skit this step and go to the Confirm step.

Follow these steps:

1. From the Import Certificates step, select the certificate entry or entries from the metadata file that you want to import.

   If you select a certificate file with invalid entries, the next dialog contains a section listing the expired entries. You cannot select these expired entries. They are displayed for your reference. If all entries in the file are invalid, the import wizard skips the certificate selection step.

   Specify a unique alias for each entry that you chose.

2. Click Next

   The Confirm dialog displays showing a table of entries.

   You can select two entries from a metadata file that have the same certificate. For SAML 1.1 and WS-Federation metadata, every entry shows Signing as the usage for the certificate because SAML 1.1 does not encrypt data.

   For SAML 2.0, each entry can show a different usage for the certificate, for example, one for signing, one for encryption. When you get to the Confirm step, the window shows a table with a single certificate entry. The certificate usage is listed as Signing and Encryption. This entry is the combination of the two entries you chose previously. This entry also uses the first alias that you specified for the certificate entry you selected.

   This situation occurs only if the same certificate was listed in the metadata file for both uses. If the file contains two separate certificates, the confirmation step shows both entries in the table.

   For example, you select two entries from the metadata file and you do not realize they are the same certificate. The first usage is Signing and you assign it the alias **cert1.** The second usage is Encryption and you assign it the alias **cert2**. When you confirm the import, you see a table titled Selected Certificate Data with an entry similar to the following entry:

Alias	**Issued To**	**Usage**
cert1	Jane Doe	Signing and Encryption

   If no usage is specified in the metadata file, then the usage defaults to Signing and Encryption.

3. Click Next to finish the configuration.

Confirm the Entity Configuration

Review the entity configuration before saving it.

Follow these steps:

1. Review the settings in the entity dialog.
2. Click Back to modify any settings from this dialog.
3. Click Finish when you are satisfied with the configuration.

A new entity is configured.

Chapter 8: Partnership Creation and Activation

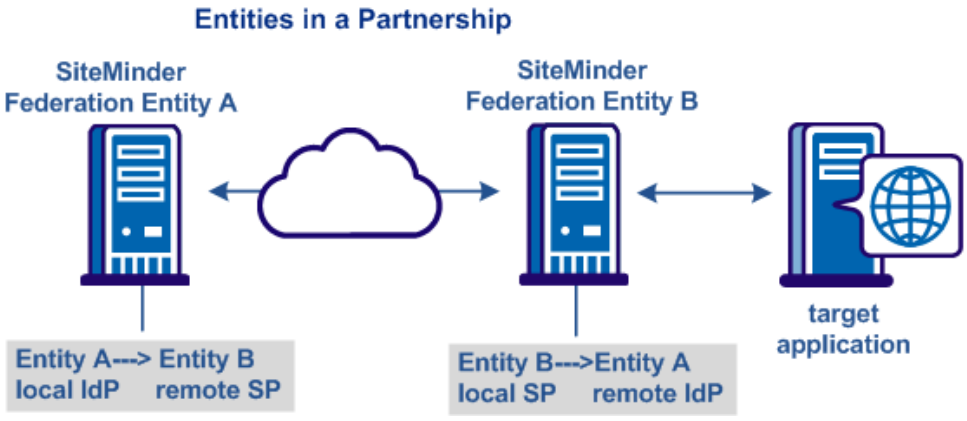
This section contains the following topics:

Partnership Creation

The main purpose of partnership federation is to establish a partnership between two organizations so they share user identity information and facilitate single sign-on (SSO). A partnership consists of two entities at different sites—one local and one remote. Either entity can assume the role of the asserting party, the side which produces assertions or the relying party, the side which consumes assertions.

If CA SiteMinder is installed at both sites, each site must define a partnership. For each local asserting party-to-relying party partnership at one site, there has to be a reciprocal local relying party-to-asserting party partnership at the partner site. For example, for the partnership configuration at Entity A, Entity A is a local Identity Provider (IdP) and Entity B is the remote Service Provider (SP). For the partnership configuration at Entity B, Entity B is the local Service Provider (SP) and Entity A is its remote Identity Provider (IdP). The perspective is based on the local entity.

The following figure shows the entity relationships for a partnership.



Note: An asserting party can have partnerships with more than one relying party and a relying party can establish partnerships with more than one asserting party.

To create a partnership, a partnership wizard takes you through the required configuration steps.

Partnership Definition

The federation partnership definition specifies which federation partner is local, and which federation partner is remote.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Partnerships.

   The Federation Partnerships dialog is displayed.
3. Click Create Partnership in the Federation Partnership List.
4. Select one of the following partnerships:
   - SAML2 IDP->SP (Identity Provider is local).
   - SAML2 SP->IDP (Service Provider is local).
   - SAML1.1 Producer ->Consumer (Producer is local).
   - SAML1.1 Consumer ->Producer (Consumer is local).
   - WSFED IP->RP (Identity Provider is local).
   - WSFED RP->IP (Resource Partner is local)

The partnership dialog opens at the first step in the partnership wizard.

Partnership Identification and Configuration

In the Configure Partnership step of the wizard, identify the partnership by naming the partnership and specifying the local and remote entities.

Note: Click Help for a description of fields, controls, and their respective requirements.

Follow these steps:

1. Enter a name for the partnership. You can use alphanumeric characters, underscores, hyphens, and periods in the name. Spaces are not allowed.
2. (Optional) Type a description.

3. Select a local entity from the local list if you have already configured an entity. If not, click Create Local Entity.
4. Select a remote entity from the remote list if you have already configured an entity. If not, click Create Remote Entity.

   Note: This step can be deferred if you are planning to create the remote entity by importing metadata later.
5. (Optional) Specify a Base URL.
6. (Optional) Enter the Skew Time in seconds.

   The skew time is the difference between the system time on the local system and the system time on the remote system. Usually, the inaccuracy of system clocks causes this condition. Determine the skew time number by subtracting the number of seconds from the current time.

   The system uses the skew time and the SSO validity duration to determine how long an assertion is valid.
7. Select one or more user directories from the Available Directories list and move them to the Selected Directories list.

   If you configure only one user directory, that directory is automatically placed in the Selected Directories list.

   Important! To use an ODBC database as a user directory, define an SQL Query scheme and valid SQL queries. These steps are necessary before you can select it as a user directory.
8. Click Next to continue through the partnership wizard. The steps of the wizard let you configure various features of a partnership, some features are required, and some are optional. The configuration details for these features are described in subsequent sections of this guide.

Note: If you are editing a partnership, you can click Get Updates next to this field to update the entity information. The latest information from the entity configuration is propagated to the partnership. However, if you edit the entity information directly from the partnership, the changes do not get propagated back to the individual entity configuration.

Editing Entities from the Partnership

You can click Get Updates next to the local and remote entity fields to update information about the entity. When you select Get Updates, the system asks to pull in the latest information from the entity.

After confirmation, the partnership you are editing is refreshed with the latest entity information. Changes are saved when you complete the partnership wizard. If you do not confirm the update, the partnership configuration remains the same.

The Entity Name identifies an entity object for in the policy store. The Entity Name must be the unique identifier because the product uses this value internally to distinguish an entity. This value is not used externally and the remote partner is not aware of this value.

If the Entity ID represents a remote partner, the value must be unique. If the Entity ID represents a local partner, it can be reused on the same system.

Note: The Entity Name can be the same value as the Entity ID, but do not share the value with any other entity.

An entity is a key component of a federation partnership. Changing an entity alters the partnership significantly; therefore, the Administrative UI does not let you replace an entity after it is in a partnership. To replace an entity, create a partnership.

To provide some flexibility within partnership configuration, you can change an entity ID because it does not identify the entity uniquely. Changing the entity ID at the partnership level does not link the partnership to another entity. The original entity in the partnership does not change. Modifications to an entity are a one-way propagation from the entity to the partnership. A change to the entity ID at the partnership does not get propagated back to the original entity.

Regard entity configurations as templates. Partnerships are created based on the entity templates so changing the partnership does not change the original entity template.

Partnership Confirmation

Review the partnership configuration before saving it.

Follow these steps:

1. Review the settings in the Confirm step of the Partnership wizard.
2. Click Modify in each group box to change any settings.
3. Click Finish when you are satisfied with the configuration.

The partnership configuration is complete.

Partnership Activation

After you configure all the required settings for a partnership, activate it to use it. You can also deactivate a partnership using the same process.

Follow these steps:

1. Select Federation, Partnership Federation, Partnerships.

   The Partnerships dialog opens.

2. From the Actions menu, select Activate or Deactivate next to the partnership of interest.

   A confirm dialog displays.

   Note: Activate is only available for a partnership in DEFINED or INACTIVE status. Deactivate is only available for a partnership in ACTIVE status.

3. Click Yes to confirm your selection.

   The status of the partnership is set and the display is refreshed.

Important! Deactivate a partnership before you modify it.

Exporting a Partnership

You can use metadata as a basis for creating remote entities and forming a partnership. Metadata makes partnership configuration more efficient because many aspects of an entity are already defined in the metadata file. The file can then be imported to create partnership or remote entity.

You do not have to complete a partnership before exporting it. You can configure a portion of the partnership and then export it.

In the Administrative UI, you can export metadata from an existing partnership entry.

Note: In the Administrative UI, you can export metadata from an existing local asserting or relying entity. When you export SAML 1.1 data, the terms used in the resulting metadata file are SAML 2.0 terms. This convention is part of the SAML specification. When you import the SAML 1.1 data, the terms are imported correctly using SAML 1.1 terminology.

When exporting from the partnership, the selected partnership is used as the basis of the export. You are not allowed to define a new partnership name. CA SiteMinder uses the name from the selected partnership.

Follow these steps:

1. Select Federation, Partnership Federation, Partnerships.

   The Partnership dialog displays.

2. Click the Action pull-down menu next to the appropriate entry in the list and select Export Metadata.

   The Export Metadata dialog opens.

3. Complete the fields on the dialog.

   If you are exporting a partnership in ACTIVE status, most of the fields are read-only. Only the Validity Duration field and the alias drop-down list are modifiable.

4. Click Export to finish.

5. A dialog prompting you to open or save the metadata file displays. You can open it to view it.

6. Save the data to an XML file on your local system.

The metadata is exported to the specified XML file.

Chapter 9: Federated User Identification for a Partnership

This section contains the following topics:

Federation Users Configuration at the Asserting Party

The Federation Users dialog is the second step in the partnership wizard when the local entity is the asserting party. This step lets you specify which users are authorized to access target resources at the remote site.

Follow these steps:

Note: Click Help for a description of fields, controls, and their respective requirements.

1. Select a user directory from the list in the Directory column of the table of the Federated Users group box.

   The pull-down list consists of one or more directory entries, depending on the number of directories you specified in the previous dialog.

2. Select the user class in the User Class column. This entry specifies a category of individual users or groups of users that can be authenticated. The options for this field depend on the type of user directory (LDAP or ODBC). Refer to the User Class tables for an explanation and example of each user class.

3. Enter a name or filter in the User Name/Filter By column. The value in this column lets the system locate the user or user group from which to authenticate federated users. This entry is dependent on the value you select for the User Class column. For examples of names and filters, see the tables at the end of this procedure.

4. (Optional) You can select Exclude for an entry to indicate that you want to exclude this user class. The default is to include all users in the directory.

   Note: An exclude criteria always takes precedence over an include criteria in case the two criteria conflict.

5. (Optional) Click Add Row to specify another user class for the same directory or another user directory.

The selection of users is complete.

Examples of User Class Entries

LDAP Examples

Use the LDAP filter syntax when specifying entries.

User Class	Valid Entry
User	Distinguished name of a user. Example: uid=user1,ou=People,dc=example,dc=com
Group	Group chosen from the list. Example: ou=Sales,dc=example,dc=com
Organization Unit	Organizational unit chosen from the list. Example: ou=People,dc=example,dc=com
Filter User Property	LDAP filter. The current user is the starting point for the search. **Example 1:** mail=user@example.com **Example 2:** (|(mail=*@.example.com)(memberOf=cn=Employees,ou=Groups,dc=example,dc=com))
Filter Group Property	LDAP filter. The current user gets authorized if they are a member of one of the groups matching the filter. The objectclasses for groups as configured in the SiteMinder registry are combined with the filter. **Example 1:** To authorize users that are members of a group with a business category of "CA Support", enter: businessCategory=CA Support **Example 2:** To authorize users that are members of a group with a description containing "Administrator" and a business category of "Administration", enter: (|(description=*Administrator*)(businessCategory=Administration)) **Note:** Not all attributes of a group work as a search criterion.

User Class	Valid Entry
Filter OU Property	LDAP filter. The current user gets authorized if they belong to an organizational unit that matches the filter. The objectclasses for organizational units as configured in the SiteMinder registry are combined with the filter. **Example 1:** To authorize users within an organizational unit with a postal code of "12345", enter: postalCode=12345 **Example 2:** To authorize users in an organizational unit with a preferred delivery method ending with "phone" and a locality of "London", enter: (|(preferredDeliveryMethod=*phone)(l=London))
Filter Any	LDAP filter. The current user gets authorized if they match the filter. **Example 1:** To authorize users with a department of "CA Support", enter: department=CA Support **Example 2:** To authorize users who are members of the group "Administrators" and have a department number of "123" or "789", enter: (&(memberof=cn=Administrators,ou=Groups,dc=example,dc=com)(|(departmentNumber=123)(departmentNumber=789)))

ODBC Examples

Use the SQL syntax when specifying queries.

User Class	Valid Entry
User	Value of the Name column for a user. The current user gets authorized if they match the entry. Example: user1
Group	Value of the Name column of a user group. The current user gets authorized if they are a member of the group that matches the query. Example: Administrators

User Class	Valid Entry
Query	A SQL SELECT statement. The current user gets authorized if they match the query. **Example 1:** With a userid of user1: Entry: SELECT * FROM SmUser Resulting query: SELECT * FROM SmUser WHERE Name = 'user1' **Example 2:** With a userid of user1: Entry: SELECT * FROM SmUser WHERE Status LIKE 'Active%' Resulting query: SELECT * FROM SmUser WHERE Status LIKE 'Active%' AND Name = 'user1' **Example 3:** With a userid of user1: Entry: SELECT * FROM SmUser WHERE Location IN ('London', 'Paris') Resulting query: SELECT * FROM SmUser WHERE Location IN ('London', 'Paris') AND Name = 'user1'

User Identification at the Relying Party

At the relying party, the partner must be able to locate a user in the local user directory. Locating the user in the user directory is the process of disambiguation. Configure the identity attribute for user disambiguation in the User Identification dialog.

The Policy Server can use one of the following methods for the disambiguation process:

- Extract the Name ID value from the assertion.
- Use the value of a specific attribute from the assertion.
- Use the value that the Xpath query obtains.

  The Xpath query locates and extracts an attribute other than the Name ID from the assertion.

After you determine which attribute is extracted from the assertion, include this attribute in a search specification. After a successful disambiguation process, the Policy Server generates a session for the user.

For SAML 2.0, you can also configure the AllowCreate feature (see page 82), which lets an asserting party create a user identifier.

Configure User Identification at the Relying Party

Configure user identification so the relying party has a method of locating a user in the local user directory.

Follow these steps:

1. Select one of the following attributes for disambiguation:
   - Name ID
   - An attribute from a previously populated drop-down list

     If the remote asserting entity was created based on metadata that contained attributes, the list is populated.
   - An attribute you enter.

     This option is most likely used when metadata is not available and the remote asserting entity does not include any attributes.
   - An Xpath query

   Click Help for the field descriptions,
2. (Optional—SAML 2.0 only) Select Allow IDP to create user identifier.

   This attribute instructs the asserting party to generate a new value for the NameID, if this feature is enabled at the asserting party. The Name ID Format entry at the asserting party must be a persistent identifier.
3. (Optional—SAML 2.0 only) Select Query parameter overrides identifier.

   This setting lets the relying party send an AllowCreate query parameter to override the value of the AllowCreate attribute configured in the authentication request. Using the query parameter instead of the identifier lets you change the value of the AllowCreate attribute without altering the partnership configuration.

   Note: For the Identity Provider to honor this query parameter setting, select the Allow IDP to create user identifier check box.
4. Specify a directory search specification for each directory listed. Two examples of search specifications are:

   LDAP Example

   uid=%s

   ODBC Example

   name=%s
5. Click Next to continue with the partnership configuration.

Employ AllowCreate for User Identification (SAML 2.0)

The SAML 2.0 AllowCreate feature is an optional setting in the User Identification configuration at the SP. Including an AllowCreate attribute in an authentication request lets an Identity Provider create a user identifier for the SP.

An SP can initiate single sign-on by sending an authentication request to the Identity Provider. As part of the request, a Service Provider can include an attribute named AllowCreate, which is set to true. The Service Provider wants to obtain an identity for the user. Upon receiving the AuthnRequest, the Identity Provider generates an assertion. The Identity Provider searches the appropriate user record for the assertion attribute serving as the Name ID. If the Identity Provider cannot find a value for the NameID attribute, it generates a unique persistent identifier for the NameID. Enable the Allow/Create feature at the Identity Provider for it to generate the identifier. The Identity Provider returns the assertion with the unique identifier back to the SP.

You can enable an AllowCreate query parameter to supersede the value of the AllowCreate attribute. Use of a query parameter lets you override the configured AllowCreate setting without deactivating, editing, and reactivating the partnership. The query parameter makes the implementation of the feature more flexible.

Chapter 10: Assertion Configuration at the Asserting Party

This section contains the following topics:

Assertion Configuration

The Assertion Configuration step of the partnership wizard defines the configuration of the following settings:

Name ID

The Name ID attribute, a required assertion attribute, identifies a user in a unique way. The Name ID format indicates the identifier type that the federated partners support. The Name ID type specifies the user profile attribute that is associated with the name ID format. The user profile attributes come from a user store or the session store.

Assertion Attributes

Servlets, web applications, or other custom applications can use attributes to display customized content or enable other custom features. When used with web applications, attributes can limit the activities of a user at the relying party. For example, an attribute variable named Authorized Amount is set it to a maximum dollar amount that the user can spend at the relying party.

Attributes are designated in an <AttributeStatement> element or an <EncryptedAttribute> element. Attributes take the form of name/value pairs. Attributes can also be made available as HTTP headers or HTTP cookies.

Note: Attributes statements are not required in an assertion.

You can configure different types of attributes for an attribute statement. The types of attributes include:

- user attributes
- DN attributes
- static data
- session attributes

  Session attributes (see page 86) are available for assertions only if they are persisted in the session store.

You can also configure an expression to transform assertion attributes. This capability is called claims transformation (see page 93).

When the relying party receives the assertion, it makes the attribute values available to applications.

Assertion Generator Plug-in

Typically, attributes come from user directory records, but an assertion can contain attributes from other sources, such as an external database or application content. You can write an assertion generator plug-in that pulls in attributes from various sources. The assertion generator plug-in is a piece of custom code that you write according to the Assertion Generator Plug-in interface.

For information about writing a plug-in, see the *Programming Guide for the Federation Java SDK*.

Configure Assertion Options

Configure assertion options at the asserting party.

Follow these steps:

1. Navigate to the Assertion Configuration step of the partnership wizard.
2. Configure the settings in the Name ID section.

   The relying party uses these values to interpret the Name ID value in the assertion.

   Depending on the selected NameID Type option, complete the entry with a proper value.

   Static attribute

   Enter any constant string in the Value field.

   User Attribute

   Enter a valid user store attribute in the Value field. For example, mail.

Session attribute

Enter a valid session store attribute in the Value field.

DN Attribute (LDAP only)

Enter a valid LDAP user directory attribute in the Value field. Also, enter a valid DN in the DN specification fields. For example, the DN attribute is cn=JaneDoe and the specification is ou=Engineering,o=ca.com.

3. (Optional - SAML 2.0 only) Select Allow Creation of User Identifier so the asserting party can create a value for the Name ID. For this feature to work, the AuthnRequest from the relying party must include an AllowCreate attribute.

   Note: If you select this option, the value of the Name ID Format value must be Persistent Identifier.

4. (Optional) Click Add Row in the Assertion Attributes table to specify one or more attributes for the assertion. Optionally, you can encrypt the attribute.

   For help filling out the table, view some assertion attribute examples (see page 86). Click Help for detailed information about each column in the attribute table.

   Note: For the LDAP user store attributes, you can add multivalued user attributes to an assertion. The Help describes how to specify multivalued user attributes.

5. (Optional) If you have written an assertion generator plug-in using the CA SiteMinder® Federation Java SDK, complete the fields in the Assertion Generator Plug-in section.

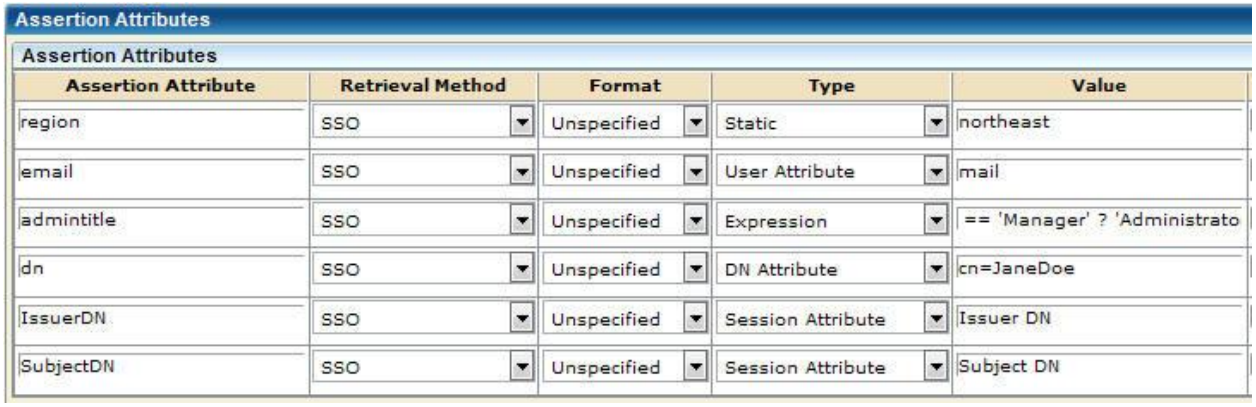
   To write a plug-in, see the *Programming Guide for the Federation Java SDK*.

6. Click Next to continue with the partnership configuration.

Assertion Attribute Configuration Examples

The following graphic shows some examples of assertion attribute entries. This screen is for a SAML 2.0 partnership. The SAML 1.1 screen is similar, but the Retrieval Method and Format columns are missing. A Namespace column exists instead.

Note: The DN Attribute example includes a DN Specification column, with the entry ou=Engineering,o=ca.com. This column is not visible in this graphic.



How To Add Session Attributes to an Assertion

The Policy Server uses the session store to persist dynamic user information after a user is authenticated. The stored information includes authentication context information, SAML attributes, third-party IdPs that authenticate users, and claims from an OAuth authentication. The Policy Server can use this information for generating user tokens or making policy decisions.

For federated single sign-on, the Policy Server can add the attributes from the session store to an assertion to customize the requested application.

Session attributes are stored for the following deployments:

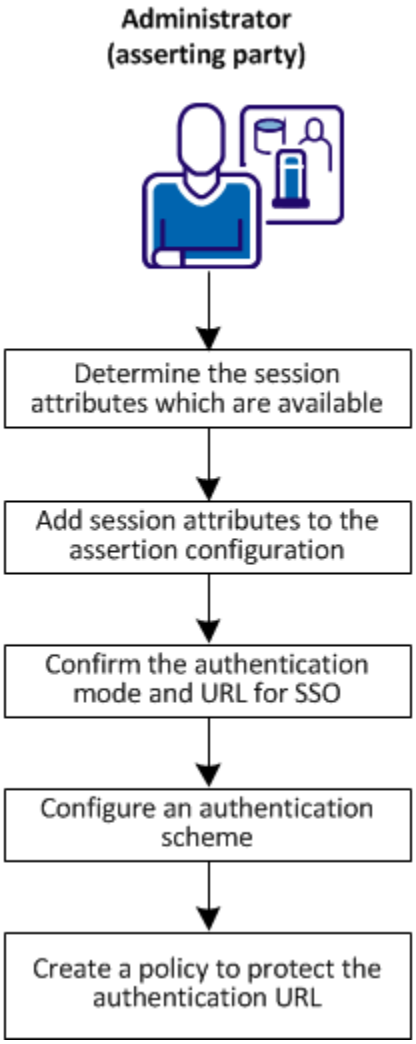
- Non-delegated authentication deployments.

  A local system or an external third party authenticates users, but the system regards it as a local authentication. Local authentication deployments require that the authentication mode is local in the single sign-on configuration. Also, an access policy must protect the authentication URL. The authentication scheme in the policy is configured to persist session attributes.

- Delegated authentication deployments

  An external third party can authenticate a user. The third-party partner returns user information, which gets stored in the session store.

The following figure shows the steps that are required to configure session attributes and add then to assertions.



Complete the following steps for session attribute support:

1. Determine which session attributes are available. (see page 88)
2. Add session attributes to the assertion configuration. (see page 88)
3. Confirm the authentication mode and URL for SSO (see page 89).
4. Configure an authentication scheme to persist session attributes. (see page 90)
5. Create a policy to protect the authentication URL. (see page 91)

Determine which Session Attributes are Available

As the federation administrator, identify the session attributes used by the partnership. Work with the authentication source, such as a database or user directory so you are familiar with the available attributes.

Add Session Attributes to the Assertion Configuration

Add session attributes to the assertion configuration. The configuration is at the asserting party, such as the IdP-to-SP partnership.

Follow these steps:

1. Log in to the Administrative UI.
2. Navigate to the Assertion Configuration step of the partnership wizard.
3. In the Assertion Attributes section, click Add Row.
4. To configure a session attribute, complete the settings in the table. For example:

   Assertion Attribute

   IssuerID

   Retrieval Method

   SSO

   Format

   Unspecified

   Type

   Session Attribute

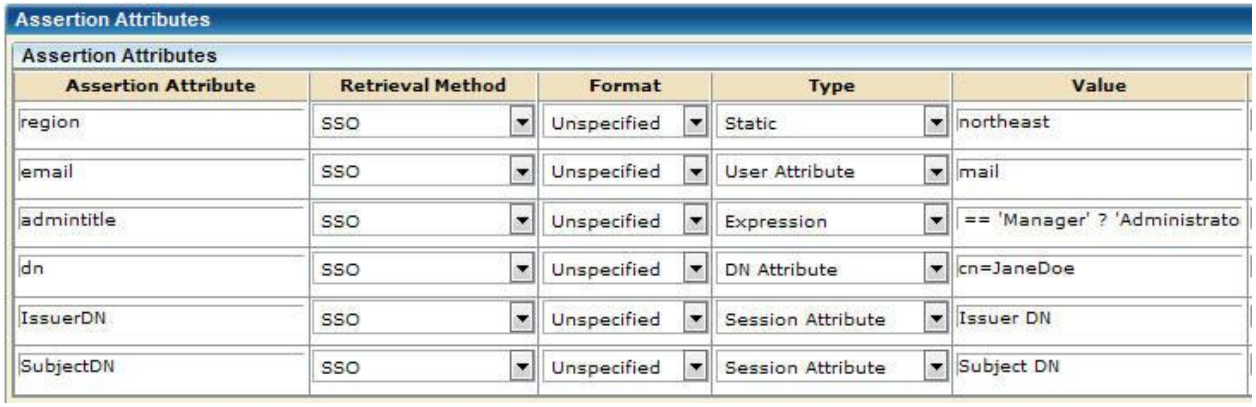
   Value

   IssuerID

   Click Help for detailed information about the attribute table.
5. Add rows for as many entries as needed.
6. (Optional). Select Encrypt to encrypt the attribute.
7. Click Next to move to the SSO and SLO step.

Session Attribute Examples in the Administrative UI

The last two entries of the following graphic show examples of session attribute entries. This screen is for a SAML 2.0 partnership. The SAML 1.1 screen is similar, but the Retrieval Method and Format columns are missing. A Namespace column exists instead.



Confirm the Authentication Mode and URL for SSO

Confirm that the partnership has the authentication mode and authentication URL set correctly.

Note: This procedure assumes that the other necessary SSO settings are configured.

Follow these steps:

1. Navigate to the SSO and SLO step of the partnership wizard.
2. In the Authentication section, verify the settings of the following fields:

   Authentication Mode

   Local

   Authentication URL

   This URL must point to the redirect.jsp file, for example:

   http://*myserver*.idpA.com/siteminderagent/redirectjsp/redirect.jsp

   myserver

   Identifies the web server with the Web Agent Option Pack or the SPS federation gateway. The redirect.jsp file is included with the Web Agent Option Pack or SPS federation gateway that is installed at the asserting party.

   Protect this resource with a policy.
3. Navigate to the Confirm step and click Finish.

Configure an Authentication Scheme to Persist Session Attributes

Configure the authentication scheme that protects the authentication URL. Enable the scheme to persist session attributes. This procedure is required for the system to store session attributes.

Follow these steps:

1. Click Infrastructure, Authentication, Authentication Schemes.
2. Click Create Authentication Scheme.
3. Verify that the Create a new object of type Authentication Scheme is selected. Click OK.

   The Create Authentication Scheme page appears.
4. Select an authentication scheme template that can persist session attributes, one which requires more information than only a user name and password.

   For example, an X.509 certificate authentication scheme requires a SubjectDN and IssuerID for the certificate. An OAuth authentication scheme requires information such as first and last name. This information can be persisted in the session store and added to an assertion.

   The authentication scheme templates that you can use are:

   - OpenID
   - OAuth
   - Any X.509 authentication template
   - Custom scheme
5. Complete the scheme-specific fields and controls.

   Click Help for the field descriptions.
6. Select Persist Authentication Session Variables in the Scheme Setup section of the dialog.
7. Click Submit to save the scheme.

Create a Policy to Protect the Authentication URL

Use the authentication scheme that persists session attributes in a policy that protects the authentication URL. When the user requests the protected resource, the policy triggers the necessary actions to authenticate the user. The system stores the credentials that the user provides as session variables.

Begin by creating a policy domain for the asserting party and assigning users. You can also modify an existing asserting party domain.

Follow these steps:

1. Click Policies, Domain, Domains.

   The Domains page appears.

2. Select the domain for the appropriate asserting party and modify it.

3. Confirm that the user directory is part of the domain. If not, add the user directory by clicking Add/Remove.

   You can select one or more user directories from the Available Members list. To select more than one member at one time, hold down the Ctrl key while you click the additional members. To select a block of members, click the first member then hold down the Shift key while you click the last member in the block.

   Note: To create a user directory and add it to the domain, click Create.

4. Click Submit.

The domain is configured.

Create a Realm and a Rule for the Authentication URL Policy

For the federation domain, create a realm and associate it with a Web Agent.

Follow these steps:

1. Click Policies, Domain, Realms.

   The Realms page appears.

2. Click Create Realm.

3. Select the domain that you want to modify, and click Next.

4. Type the name and a description of the realm.

   Specify a name that indicates the realm is for an SSO authentication URL.

5. Select an Agent by clicking Lookup Agent/Agent Group.

6. Select the appropriate Web Agent and click OK.

7. Specify the Resource Filter for the redirect.jsp. For example:

   /siteminder/redirectjsp/redirect.jsp

8. Complete the remaining fields:

   Default Resource Protection

   Protected

   Authentication Scheme

   Select the authentication scheme that you configured to protect the authentication URL. This scheme is the one you configured to persist session attributes.

9. Create a rule in the Rules section.
   a. Specify a name for the rule.
   b. Accept the defaults for the remaining settings.
10. Skip the other configuration options.
11. Click Finish.

The realm and rule configuration is complete.

Complete the Authentication URL Policy

Create a policy that protects the authentication URL. The policy components work together and protect the resource.

After you create the policy, add users and rules.

Follow these steps:

1. Click Policies, Domain, Domains.
2. Search for the domain.

   A list of domains that match the search criteria appears.

3. Select the domain for the asserting party.
4. Click Modify.
5. Click the Policies tab.

   The Policies page appears.

6. Click Create.
7. Enter a name and a description for the policy.

8. Add individual users, user groups, or both from the Users tab. The users are members of the user directory that is associated with the domain.

   From within each user directory group box, choose Add Members, Add Entry, or Add All. Depending on which method you use, a dialog box opens enabling you to add users.

   Note: If you select Add Members, the User/Groups pane opens. The individual users are not displayed automatically. To find a specific user within one of the directories, use the search utility.

   You can edit or delete a user or group by clicking the right arrow (>) or minus sign (-), respectively.

9. Add a rule from the Rules tab.
10. Select the rule that you created for the authentication URL and click OK.

    You are not required to configure a response for the rule.

11. Click Submit to complete the configuration.

The policy configuration is complete.

The assertion attribute, single sign-on, and policy configuration work together to make session attribute available for assertions.

How to Configure Claims Transformation at the Asserting Party

Claims transformation manipulates claims during a federated single sign-on transaction. Claims, also known as attributes, help customize the attributes and improve the user experience at a partner.

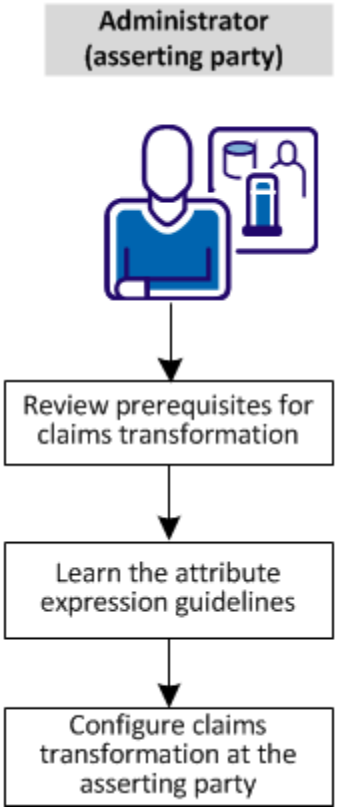
Modifying assertion attributes lets the relying party adapt user information so a target application can use it. For example, claims transformation can associate roles at different partners in different domains. In one domain, a user is an engineering manager and belongs to a group named EngineerAdmins. However, the relying party identifies the same role as DevelAdmins. The asserting party alters the role attribute before issuing the assertion. The user is now identified with the DevelAdmins role, which the relying party application can understand.

Claims transformation occurs at the local asserting party during the assertion generation process. You configure the feature on a per-partnership basis. An assertion can be modified whether a local or remote party generates the assertion. Claims are transformed based on an expression that you configure for the partnership. The expression relies on user information from the user store and the CA SiteMinder session store.

The software can perform three different modifications to assertion attributes:

- **Transformation:** Changing the value of an assertion attribute to a different value.
- **Addition:** Adding an assertion attribute if it does not exist already.
- **Deletion:** Deleting an assertion attribute on a conditional basis.

The following figure shows the configuration steps:



To set up claims transformation, perform the following steps:

1. Review the prerequisites for claims transformation. (see page 95)
2. Learn the attribute expression guidelines. (see page 95)
3. Configure claims transformation at the asserting party. (see page 97)

Prerequisites for Claims Transformation

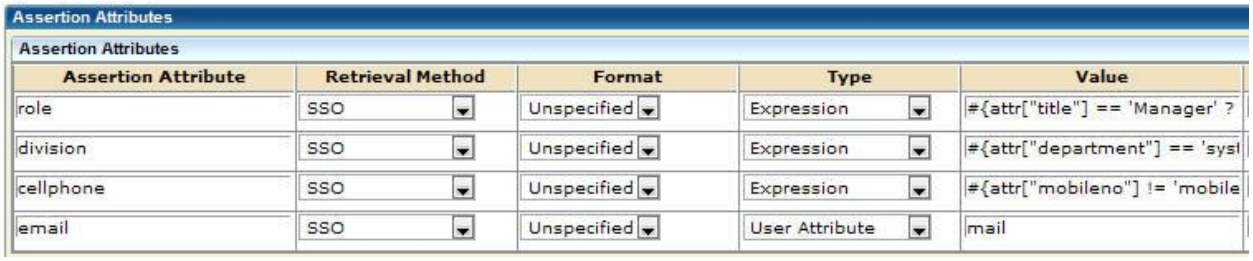
Before you configure claims transformation, consider the following prerequisites:

- Be familiar with the user store and session store attributes available.
- Determine which attributes the relying party expects to receive in an assertion.
- Be familiar with Java Unified Expression Language (JUEL), an open source version of the Unified Expression Language.

Learn the Attribute Expression Guidelines

Expressions are rules that instruct the software how to manipulate assertion attributes. The expression directs the software to modify, add, or delete assertion attributes. You construct expressions using the Java Unified Expression Language (JUEL). A JUEL expression evaluator examines the configured expressions and generates the resulting assertion attributes.

Define expressions in the Assertion Attributes table of the Administrative UI. Access this table by navigating to the Assertion Configuration step of the partnership wizard. This table is shown in the following figure:



Enter expressions in the Value column of the assertion attributes table. All attributes in an expression are user store or session store attributes.

Typically, the expression operates on a conditional basis. If the condition is met, the designated claims modification occurs. For example, an incoming assertion contains the "role" attribute. The expression to modify the "role" assertion attribute is:

#{attr["title"] == 'manager' ? 'administrator' : attr["title"]}

The first part of the expression **#{attr["title"] == 'manager'** tells the software to determine whether the logged-in user has the title "manager." The lookup is done in the user directory. If this condition is met, the second part of the expression, **? 'administrator' :** assigns the value "administrator" to the role assertion attribute. If the condition is not met, the last part of the expression, **attr["title"]}** indicates that the value of the user attribute "title" remains "manager." The value "manager" is assigned to the assertion attribute "role."

Note: You can use static values in an expression in place of the syntax attr["title"], such as 'administrator' in the previous example.

The example assumes the "role" attribute is already in the assertion. Therefore, the expression is a transformation of an existing attribute. If "role" is not part of the assertion, the software adds the role attribute to the assertion.

Expression Syntax

Construct expressions using the proper syntax:

- Represent a user store attribute with the string **attr**["*attribute_name*"].
- Represent a session store attribute with the string **session_attr**["*attribute_name*"].
- Delete a claim using the argument 'DELETE'.

Use lower-case text for the **attr** and **session_attr** prefixes. Attribute names are not case-sensitive.

Additionally, be aware of these conditional JUEL operators:

Operator	Meaning
conditional value ? value1 : value2	The conditional value resolves to either value1 or value 2.
!=	Does not equal
==	Equals

Important! The attribute in the expression must be available in the user directory or the session store. If an attribute is incorrect, the system simply includes blanks for corresponding attributes. The assertion generation does not fail.

For more examples of expressions, read the section Configure claims Transformation at the Asserting Party (see page 97).

Configure Claims Transformation at the Asserting Party

Define expressions at the partnership level. The result of these expressions modifies, adds, or deletes attributes from assertions. After the rules are defined, the assertion is modified and sent to the relying party. If you do not configure claims transformation, assertion attributes are passed "as is" to the relying party.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnerships.
3. Select a partnership that you want to modify. Eligible partnerships include:
   - Local Producer to Remote Consumer
   - Local IdP to Remote SP
   - Local IP to Remote RP
4. Navigate to the Assertion Configuration step in the partnership wizard.

   In the Assertion Attributes section, click Add Row.
5. Pay particular attention to the following fields in the row. Click Help for detailed descriptions of each field.

   Assertion Attribute

   Enter an assertion attribute. All values in this column are assertion attributes. An attribute that is already in the assertion remains in the assertion, but it is set to a new value based on the configured expression. The attribute is removed from the assertion only if you configure a DELETE expression.

   Retrieval Method

   Keep the default, SSO.

   Format

   Designates the format for the attribute being added to the assertion. The format options vary depending on the SAML profile for the entity.

   Type

   Expression

   Always use this value for claims transformation.

Value

Enter an expression that reflects how you want the assertion attribute is modified.

Review the guidelines about constructing claims expressions (see page 95) and the following examples:

- Transform a claim in an assertion. (see page 98)
- Add a claim to an assertion (see page 100).
- Delete a claim from an assertion (see page 101).

6. (Optional for SAML 2.0 and WSFED with token type SAML 2.0). To encrypt assertion attributes, select Encrypt. The asserting party encrypts the assertions using the certificate that is specified in the partnership configuration.

   The relying party decrypts the assertion attributes using the private key that is associated with the certificate.

7. Add as many rows as you like for the assertion attributes you want to configure.

Claims transformation is implemented based on the configured entries in the partnership.

Transform a Claim in an Assertion

Transforming a claim changes an assertion attribute value to another value.

Note: These examples only show entries for Assertion Attribute, Type, and Value.

Transformation Example 1

The following example assumes the "title" attribute is already in the assertion. The table indicates the user attributes in the user store.

User Directory Attributes	Attribute Value
role	admin
admintitle	SeniorAdmin
supertitle	SuperUser

Transform the value of the existing title attribute using the following configuration:

Assertion Attribute

title

Type

Expression

Value

#{attr[“role”] == ‘admin’ ? attr[“admintitle”] : attr[“supertitle”]}

Result: The expression is conditional based on the "role" user attribute being set to "admin." Assuming this condition is met, the assertion attribute "title" is set to the value SeniorAdmin, the "admintitle" attribute. If the role is something other than "admin," the "title" attribute becomes SuperUser, the value of the "supertitle" attribute.

Transformation Example 2

The following example assumes that the ContactNo attribute is already in the assertion.

User Directory Attributes	Attribute Value
homephone	555-3344
mobile	555-8888

Transform the value of the existing title attribute using the following configuration:

Assertion Attribute

ContactNo

Type

Expression

Value

#{attr["homephone"] == '555-3344' ? attr["mobile"] : attr["homephone"]}

Result: The expression is conditional on the logged-in user having the "homephone" user attribute set to 555-3344. Assuming this condition is met, the assertion attribute is set to 555-8888, the value of the "mobile" attribute. If the condition is not met, the "homephone" value does not change.

Note: To configure an expression that uses session attributes, replace attr["*attribute_name*"] with session_attr[“*attribute_name*”]. For example:

#{session_attr[“att1”] == ‘admin’ ? session_attr[“attr2”] : attr[“attr3”]}

Add a Claim to an Assertion

You can add an assertion attribute that is not already present.

Addition Example 1

The following example assumes the "title" assertion attribute is *not* in the assertion.

User Directory Attribute	Attribute Value
role	admin
admintitle	director
supertitle	executive

The following configuration adds the title attribute to the assertion.

Assertion Attribute

title

Type

Expression

Value

#{attr[“role”] == ‘admin’ ? attr[“admintitle”] : attr[“supertitle”]}

Result: The expression is conditional on the logged-in user having the "role" attribute set to admin. Assuming this condition is met, the assertion attribute "title" is added to the assertion and set to the value "director," the "admintitle" attribute value. If the role is something other than "admin," the assertion attribute "title" is added but the value becomes "executive," the value of the "supertitle" attribute.

Addition Example 2

The following example assumes the "smtitle" assertion attribute is *not* in the assertion.

User Directory Attribute	Attribute Value
title	manager

Assertion Attribute

smtitle

Type

Expression

Value

#{attr["title"] == 'manager' ? 'federation administrator' : attr["title"]}

Result: If the logged-in user has the title of "manager," add "smtitle" to the assertion and set its value to "federation administrator." You can enter any static value after the question mark instead of using the syntax attr["attribute_name"]. In this example, the static value is federation administrator,

Note: To configure an expression that uses session attributes, replace attr["*attribute_name*"] with session_attr[“*attribute_name*”]. For example:

#{session_attr[“att1”] == ‘admin’ ? session_attr[“attr2”] : attr[“attr3”]}

Delete a Claim from an Assertion

You can delete an assertion attribute.

Deletion Example 1

Delete the admintitle and supertitle assertion attributes by configuring two entries.

User Directory Attribute	Attribute Value
role	admin or superuser

User Directory Attribute	Attribute Value
title	administrator
su	superuser

Assertion Attribute

admintitle

Type

Expression

Value

#{attr[“role”] == ‘superuser’ ? ‘DELETE’ : attr[“title”]}

Result: The expression string is conditional based on the "role" user attribute. If the logged-in user has the role of superuser, delete the assertion attribute "admintitle." If the role is not superuser, set the title assertion attribute to the value of administrator, the value of the title user directory attribute.

Assertion Attribute

supertitle

Type

Expression

Value

#{attr[“role”] == ‘admin’ ? ‘DELETE’ : attr[“su”]}

Result: The expression string is conditional on the "role" user attribute. If the logged-in user role is "admin," delete the assertion attribute "supertitle." If the role is not "admin," set the supertitle assertion attribute to the value of superuser, the value of the su user directory attribute.

Deletion Example 2

The following example combines an addition and a deletion using one expression.

User Directory Attribute	Attribute Value
title	manager

Assertion attribute

ManagerName

Type

Expression

Value

#{attr["title"] != 'Manager' ? attr["manager"] : 'DELETE'}

Result: If the logged-in user does *not* have the user attribute title "manager," add the ManagerName attribute to the assertion. However, if the logged-in user title is manager, delete the ManagerName attribute, assuming it is part of the assertion.

Note: To configure an expression that uses session attributes, replace attr["*attribute_name*"] with session_attr[“*attribute_name*”]. For example:

#{session_attr[“att1”] == ‘admin’ ? session_attr[“attr2”] : attr[“attr3”]}

Customize Assertion Content

Implement the AssertionGeneratorPlugin Interface

The first step in creating a custom assertion generator plug-in is to implement the AssertionGeneratorPlugin interface. The following requirements apply to the implementation class:

- The implementation must provide a public default constructor method that contains no parameters.
- The implementation must be stateless, so that many threads can use a single plug-in class.
- The implementation must include a call to the customizeAssertion methods. You can overwrite the existing implementations of these methods as your requirements dictate. See the sample programs.
- The syntax requirements and use of the parameter string that is passed into the customizeAssertion method is the responsibility of the custom object.

Note: The folder *federation_sdk_home*\sample\com\ca\federation\sdk\plugin\sample includes two sample implementation classes.

Deploy an Assertion Generator Plug-in

After you have coded your implementation class for the AssertionGeneratoPlugin interface, compile it and verify that CA SiteMinder® Federation can find your executable file.

To deploy the assertion generator plug-in

1. Compile the assertion plug-in code in one of the following ways:
   - If you are using a sample plug-in, use the build script for your platform to compile the plug-in. The build scripts are installed in the directory *federation_sdk_home*\sample. The build scripts are:

     Windows: build_plugin.bat

     UNIX: build_plugin.sh

     A compiled sample plug-in, fedpluginsample.jar, is in the directory *federation_sdk_home*\jar.
   - If you write your own plug-in, include the smapi.jar when you compile your plug-in.
2. In the JVMOptions.txt file, modify the -Djava.class.path value so it includes the classpath for the plug-in. Locate the JVMOptions.txt file in the directory *federation_install_dir*\siteminder\config.

   You can place the plug-in jar in any directory and have the JVMOptions.txt file point to it. To use the sample plug-in, modify the classpath to point to fedpluginsample.jar; however, do not modify the classpath for smapi.jar.

   Note: To use Apache Xerces or Xalan in your plug-in, use the Xerces or Xalan binary files installed with CA SiteMinder® Federation. The binaries are not installed with the CA SiteMinder® Federation SDK. Using these files is necessary for compatibility reasons.
3. Restart the CA SiteMinder® Federation services.

   Restarting the services helps ensure that CA SiteMinder® Federation uses the latest version of the assertion generator plug-in.

Enable the Assertion Generator Plug-in

After writing an assertion generator plug-in and compiling it, you enable the plug-in by configuring settings in the CA SiteMinder® Federation UI. The UI parameters let CA SiteMinder® Federation know where to find the plug-in.

Do not configure the plug-in settings until you deploy the plug-in (see page 104).

To enable the Assertion Generator plug-in

1. Log on to the Administrative UI.
2. Navigate to the Assertion Configuration step of the Partnership wizard for the partnership you want to modify.
3. Enter values for the Assertion Generator Plug-in settings that follow:

   Plug-in Class

   Specifies the Java class name of the plug-in. Enter a name. This plug-in is invoked at run time.

   Example: com.mycompany.assertiongenerator.AssertionSample

   The plug-in class can parse and modify the assertion, and then return the result to CA SiteMinder® Federation for final processing. Specify an Assertion Generator plug-in for each relying party. A compiled sample plug-in is included in the SDK. You can view complied sample assertion plug-ins in the directory *federation_sdk_home*/jar.

   Note: You can also view the source code for the CA SiteMinder® Federation sample plug-ins in the directory *federation_sdk_home*\sample\com\ca\federation\sdk\plugin\sample.

   Plug-in Parameter

   (Optional). Specifies the string that CA SiteMinder® Federation passes to the plug-in as a parameter at run time. The string can contain any value; there is no specific syntax to follow.

   The plug-in interprets the parameters that it receives. For example, the parameter could be the name of an attribute or the string can contain an integer that instructs the plug-in to do something.

Reference information (method signatures, parameters, return values, data types), and the constructor for UserContext class and the APIContext class, are in the *Javadoc Reference*. Refer to the AssertionGeneratorPlugin interface in the Javadoc.

Chapter 11: Single Sign-on Configuration

This section contains the following topics:

Single Sign-on Configuration (Asserting Party)

To specify how assertions are delivered to a relying party, configure single sign-on at the asserting party.

The procedure that follows offers the basic steps to enable single sign-on. Details about all the configurable features in the sign-on dialog are described in subsequent topics and in the Administrative UI help.

Follow these steps:

1. Begin at the appropriate step in the partnership wizard.

   SAML 1.1

   Single Sign-On

   SAML 2.0

   SSO and SLO

   WSFED

   Single Sign-on and Sign-Out

   Any values that are defined during the creation or import of the remote relying party are filled in.

2. Complete the fields in the Authentication section, noting the following information:

   - If you select Local for the Authentication Mode field, enter a URL for the Authentication URL that points to the redirect.jsp file. For example:

     http://webserver1.example.com/affwebservices/redirectjsp/redirect.jsp

     In this example, webserver1 identifies the web server with the Web Agent Option Pack. The redirect.jsp file is included with the Web Agent Option Pack installed at the Identity Provider site.

     Important! Protect the Authentication URL (see page 57) with an access control policy. Configure the realm, rule, and policy. To add session information to the assertion, enable the Persist Authentication Session Variables check box.

   - If you select Delegated as the Authentication mode, configure the additional fields. Learn more about delegated authentication (see page 181).

3. Complete the Authentication Class field (SAML 1.1 and 2.0 only). Supply a static URI for this field. Additionally, for SAML 2.0 only, the software can automatically detect an authentication class. The URI is placed in the AuthnContextClassRef element in the assertion to describe how a user is authenticated.

4. Complete the fields in the SSO section. These settings let you control the following features:

   - single sign-on binding
   - assertion validity

     The SSO Validity Duration and the Skew Time determine when the assertion is valid. To understand how these settings work together, read the information about assertion validity (see page 113).

   For SAML 2.0, you can configure these features:

   - Initiation of single sign-on from which partner
   - SP session validity
   - SP session duration
   - User consent to share identity information with the SP

   Click Help for the field descriptions.

5. Specify the URL for the assertion consumer service or security token service. This remote relying party service consumes and processes assertions.

   Your partner must supply this URL to you.

6. If you selected HTTP-Artifact as the SAML binding, configure the back channel settings (see page 115).

7. (Optional). For SAML 2.0, you can do the following tasks:

   - Enable IDP Discovery Profile (see page 128).
   - Specify status redirect URLs (see page 112) for specific HTTP errors.

More information:

SAML 2.0 Entities Allowed to Initiate Single Sign-on (see page 113)
Status Redirects for HTTP Errors (SAML 2.0 IdP) (see page 112)
Legacy Artifact Protection Type for the HTTP-Artifact Back Channel (see page 109)

Authentication Mode for Partnership Federation

Partnership federation lets you define the authentication mode for federated single sign-on.

- **Local authentication mode**

  Local authentication primarily happens at the local federation system. For local authentication, you can select Basic or Forms as the authentication schemes. These options are the only two methods available locally.

  You can also select local for the authentication mode when an external third party authenticates a user. When the third party passes back the user information, the user information gets stored in the session store for later use in assertions.

- **Delegated authentication mode**

  Delegated authentication forwards the authentication task to a third-part web access management (WAM) system. The method by which the third party authenticates a user depends on the authentication schemes the third party supports. After the third-party WAM authenticates the user, it sends the federated user identity back to CA SiteMinder.

Legacy Artifact Protection Type for the HTTP-Artifact Back Channel

For HTTP-Artifact single sign-on, you can select the legacy option for the Artifact Protection Type field. The legacy option indicates that you are using the legacy method of protecting the back channel to the artifact service at the asserting party.

To implement the legacy method of protection:

- Add the Web Agent that protects the FWS application to the Agent group FederationWebServicesAgentGroup.
  - For ServletExec, this Agent is on the web server where the Web Agent Option Pack is installed.
  - For an application server, such as WebLogic or JBOSS, this Web Agent is installed where the application server proxy is installed. The Web Agent Option Pack can be on a different system.
- Enforce the policy that protects the artifact service. To enforce the policy, you indicate which asserting party-to-relying party partnerships are permitted access to the artifact service.

Follow these steps: to add a web agent to an agent group

1. Log in to the Administrative UI.
2. Select Infrastructure, Agents, Create Agent.
3. Specify the name of the Web Agent in your deployment. Click Submit.
4. Select Infrastructure, Agent Groups.
5. Select the FederationWebServicesAgentGroup entry.

   The Agent Groups dialog opens.
6. Click Add/Remove and the Agent Group Members dialog opens.
7. Move the web agent from the Available Members list to the Selected Members list.
8. Click OK to return to the Agent Groups dialog.
9. Click Submit then click Close to return to the main page.

Follow these steps: to enforce the policy that protects the retrieval service

1. In the Administrative UI, configure the partnership using the legacy method for the artifact protection type.
2. Activate this partnership.
3. Select Policies, Domain, Domain Policies.

   A list of available domain policies displays.
4. Edit the appropriate artifact service policy by selecting the pencil icon.

   SAML 1.1

   FederationWSAssertionRetrievalServicePolicy

   SAML 2.0

   SAML2FWSArtifactResolutionServicePolicy

   Note: The supplied policies are default policies. You can use any policy that you created to protect the artifact service.
5. Go to the Users tab.

   The federation custom user stores display in the User Directories section.
6. Click Add Members for the user store you want to modify:

   SAML 1.1

   FederationWSCustomUserStore

   SAML 2.0

   SAML2FederationCustomUserStore

7. Select the partnerships for which you configured legacy artifact protection.

   Examples:

   - If the SAML 1.1 partnership is named Acme, select affiliate:affiliate:Acme
   - If the SAML 2.0 partnership is named Demo, select affiliate:samlsp:Demo

8. Click OK.

The partnership for HTTP-Artifact single sign-on now allows the access to the artifact service so the relying party can retrieve the assertion.

Single Sign-on Configuration (Relying Party)

To configure single sign-on at the relying party, specify the SAML binding and the other related SSO settings.

At the relying party, the system uses the skew time for the partnership to determine whether the assertion it receives is valid. To understand how the system uses the configured skew time, read more about assertion validity (see page 113).

The procedure that follows offers the basic steps to enable single sign-on. Details about all the configurable features in the sign-on dialog are described in subsequent topics and in the Administrative UI help.

Follow these steps:

1. Begin at the appropriate step in the partnership wizard.

   SAML 1.1

   Single Sign-On

   SAML 2.0

   SSO and SLO

   WS-Federation

   Single Sign-On and Sign-Out

2. Configure the settings in the SSO section of the dialog. These settings let you control the single sign-on binding.

   Click Help for the field descriptions.

   For SAML, configure the HTTP-Artifact or the HTTP-POST profile. If the relying party initiates single sign-on, it includes a query parameter in the request. This query parameter indicates the SSO binding to use. If no binding is specified, the default is POST. If the asserting party initiates single sign-on, the asserting party indicates the binding in use for that particular transaction.

3. (Optional). For SAML 2.0, you can configure these settings:
   - Remote SSO Service URLs
   - Remote SOAP Artifact URLs
   - Initiation of single sign-on from which partner

     If a third-party IdP is authenticating a consumer user with no user record at the host, SSO is initiated at the SP.
   - User consent requirement
4. If you select the HTTP-Artifact profile, configure the authentication method for the back channel in the Back Channel section of the dialog.
5. For the remaining settings, accept the defaults.

The basic settings for single sign-on are complete. Other settings are available for SSO. Click Help for the field descriptions.

Status Redirects for HTTP Errors (SAML 2.0 IdP)

For the Identity Provider, you can configure how CA SiteMinder redirects a user when an HTTP 500, 400, or 405 error occurs. For example, a 403 error can occur because the URL in a request points to the wrong target. If this error occurs, the user is sent to the specified URL for further processing.

Select the redirect options as follows:

1. Navigate to the Status Redirect URL section of SSO and SLO dialog.
2. In the Status Redirect URL section, select the check box for the error conditions that prompt a redirect.
3. Enter the destination URL where CA SiteMinder redirects the user.
4. For each URL, select the redirect method, 302 No Data or HTTP Post.

Redirect handling is configured.

SAML 2.0 Entities Allowed to Initiate Single Sign-on

For SAML 2.0 partnerships, you can determine whether the IdP or the SP or both can initiate single sign-on. You can configure which transactions are allowed at each side of the partnership.

Consider how restricting the initiation of a transaction can impact other single sign-on features, such as exchanging user authentication context information.

Follow these steps:

1. Log in to the Administrative UI.
2. Select the SAML 2.0 partnership you want to edit.
3. Navigate to the SSO and SLO step of the partnership wizard.
4. In the Transactions Allowed field, select an option from the pull-down menu.
5. Skip to the Confirm step of the wizard and save your changes.

Assertion Validity for Single Sign-on

For single sign-on, the values of the Skew Time and the SSO Validity Duration determine how long an assertion is valid. The Policy Server applies the skew time to the generation and consumption of assertions. In the assertion document, the NotBefore and NotOnOrAfter values represent the beginning and end of the validity interval.

At the asserting party, the Policy Server sets the assertion validity. The Policy Server determines the beginning of the validity interval by taking the system time when the assertion is generated. The software sets the IssueInstant value in the assertion from this time. The Policy Server then subtracts the skew time value from the IssueInstant value. The resulting time becomes the NotBefore value.

NotBefore=IssueInstant - Skew Time

To determine the end of the validity interval, the Policy Server adds the Validity Duration value and the skew time to the IssueInstant value. The resulting time becomes the NotOnOrAfter value.

NotOnOrAfter=Validity Duration + Skew Time + IssueInstant

Times are relative to GMT.

For example, an assertion is generated at the asserting party at 1:00 GMT. The skew time is 30 seconds and the validity duration is 60 seconds, making the assertion validity interval between 12:59:30 GMT and 1:01:30 GMT. This interval begins 30 seconds before the time the assertion was generated and ends 90 seconds afterward.

At the relying party, the Policy Server performs the same calculations as it does at the asserting party to determine if the assertion it receives is valid.

Calculating Assertion Validity when CA SiteMinder is at Both Sides of the Partnership

The total time the assertion is valid is the sum of the SSO validity duration plus two times the skew time. The equation is:

Assertion Validity = 2x Skew Time (asserting party) + SSO Validity Duration + 2x Skew Time (relying party)

The initial part of the equation (2 x Skew Time + SSO Validity Duration) represents the validity window at the asserting party. The second part of the equation (2 x Skew Time) represents the skew time of the system clock at the relying party. You multiply by 2 because you are accounting for the NotBefore and the NotOnOrAfter ends of the validity window.

Note: For the Policy Server, the SSO Validity Duration is only set at the asserting party.

Example

Asserting Party

The values at the asserting party are as follows:

IssueInstant=5:00PM

SSO Validity Duration=60 seconds

Skew Time = 60 seconds

NotBefore = 4:59PM

NotOnOrAfter=5:02PM

Relying Party

The relying party takes the NotBefore and NotOnOrAfter values that is receives in the assertion then applies its skew time to calculate new values.

Skew Time = 180 seconds (3 minutes)

NotBefore = 4:56PM

NotOnOrAfter=5:05PM

Based on these values, the calculation for the total assertion validity window is:

120 seconds (2x60) + 60 seconds + 360 seconds (2x180) = 540 seconds (9 minutes).

Session Validity at a Service Provider

You can manage the duration of the authentication session at the Service Provider. The SessionNotOnOrAfter attribute is an optional attribute that the IdP can include in the <AuthnStatement> of an assertion. The configuration for session validity is done at the IdP.

Note: The SessionNotOnOrAfter parameter is different from the NotOnOrAfter parameter, which determines how long the assertion is valid.

A third-party SP can use the value of the SessionNotOnOrAfter to set its own timeout values, helping to ensure that sessions are not too short. If a user session becomes invalid, the user has to reauthenticate at the Identity Provider.

Important! If CA SiteMinder is acting as an SP, it ignores the SessionNotOnOrAfter value. Instead, a CA SiteMinder SP sets session timeouts from the realm timeout that corresponds to the SAML authentication scheme protecting the target resource.

Follow these steps:

1. Log in to the Administrative UI.
2. Select the IdP->SP partnership you want to modify.
3. Navigate to the SSO and SLO step.
4. In the SSO section, select the option for the SP Session Validity Duration. If you select the customize option, you can select several options.

   Click Help for the field descriptions.
5. Select the Confirm step after you complete your changes and click Finish.

Back Channel Authentication for Artifact SSO

Artifact single sign-on requires the relying party to send an artifact to the asserting party to retrieve the assertion. The asserting party uses the artifact to retrieve the correct assertion and returns the assertion to the relying party over a back channel.

You can require an entity to authenticate to access the back channel. The back channel can also be secured using SSL, though SSL is not required.

Securing the back channel using SSL involves:

1. Enabling SSL.

   SSL is not required for Basic authentication but you can use Basic over SSL. SSL is required for Client Cert authentication.

2. Configuring an incoming or outgoing back channel for the SAML 2.0 communication exchange. The direction you configure depends on the role of the local entity.

   Configuring separate channels is supported only for SAML 2.0. The back channel configuration for SAML 1.1 artifact single sign-on uses a single configuration for each partnership. CA SiteMinder uses the correct direction automatically (incoming for a local producer and outgoing for a local consumer).

   Select which direction to configure for SAML 2.0 single sign-on based on the entity you are configuring.

   - The local asserting party uses the incoming channel.
   - The local relying party uses the outgoing channel.

   Note: You can configure an incoming and outgoing back channel; however, a channel can have only one configuration. If two services use the same channel, these two services use the same back channel configuration. For example, if the incoming channel for a local asserting party supports HTTP-Artifact SSO and SLO over SOAP, these two services must use the same back channel configuration.

3. Choosing the type of authentication for the relying party to gain access across the protected back channel. The authentication method applies per channel (incoming or outgoing).

   The options for back channel authentication are:

   - Basic
   - Client Cert
   - NoAuth

   The Administrative UI help describes these options in detail.

   Important! The authentication method for the incoming back channel must match the authentication method for the outgoing back channel on the other side of the partnership. Agreeing on the choice of authentication method is handled in an out of band communication.

SAML 2.0 Attribute Query Support

A CA SiteMinder IdP supports the SAML 2.0 Assertion Query/Request profile and can respond to attribute queries. The IdP also extends the profile functionality by accepting queries for attributes not in the assertion or in the metadata. When the IdP receives an attribute query, the IdP first checks its user directory to find the attributes. If the attributes are not found, the Policy Server checks the session store. The session store can hold attributes from external Identity Providers, attributes collected from advanced authentication schemes, and other sources.

Note: Only the CA SiteMinder IdP supports the query profile. A CA SiteMinder SP as an attribute requester is only supported for the proxied attribute query feature (see page 119).

The IdP has all the user attributes that an SP can request in its metadata. An SP can obtain these attributes in two ways:

- Extract the set of attributes that are sent in an assertion.

  The Identity Provider assertion configuration determines the set of attributes included. Defining a subset of all the attributes limits the number of attributes to the most essential, which reduces processing overhead.

- Import the IdP metadata.

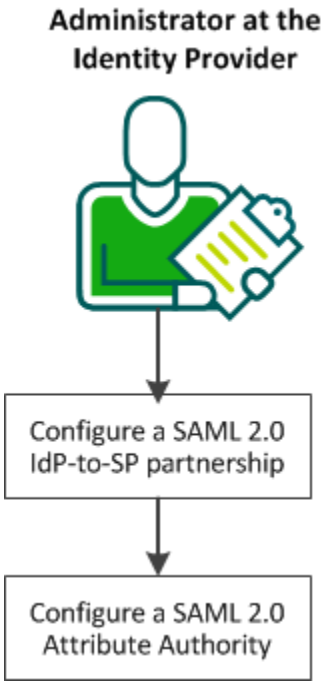
In addition to the attributes in the metadata, an SP can require attributes that are not in the assertion or in the metadata. To retrieve other attributes, the SP sends an attribute query to the IdP.

The query request profile employs two entities:

- SAML Attribute Authority
- SAML Attribute Requester

A CA SiteMinder IdP can only act as an Attribute Authority. A CA SiteMinder SP cannot be the Attribute Requester.

The following graphic shows the configuration steps for an Attribute Authority.



Complete the the following steps:

- Configure or modify an IdP-to-SP partnership. (see page 118)
- At the Identity Provider, configure a SAML 2.0 Attribute Authority (see page 118).

If CA SiteMinder is at both sides of the partnership, you cannot use the Assertion Query/Response profile.

Configure the Partnership for Attribute Query Support

For the IdP to respond to attribute queries, an IdP-to-SP partnership must exist. You can create a partnership or modify an existing partnership.

The steps for creating a partnership include:

1. Create the SAML 2.0 IdP and SP entities (see page 61).
2. Configure a connection to a user directory for the partnership (see page 55).
3. Create a SAML 2.0 IdP-to-SP partnership (see page 71).
4. Configure a SAML 2.0 Attribute Authority (see page 118).

These steps are detailed throughout this guide.

Configure the SAML 2.0 Attribute Authority

You can configure an IdP to serve as an Attribute Authority.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Partnerships.
3. Select the IdP-to-SP partnership that you want to modify or create a new one.
4. Navigate to the SSO and SLO step of the partnership wizard.
5. Select Enable in the Attribute Service section of the dialog.
6. Enter a number of seconds for the Validity Duration.
7. (Optional) Specify whether to require that the attribute query is signed, and the signing requirements for attribute assertions and responses.
8. Enter the search specifications for the appropriate user directory name space in the User Lookup section. The Attribute Authority uses this search specification to disambiguate the user.

   An example for an LDAP user directory is uid=%s. At least one search specification is required.

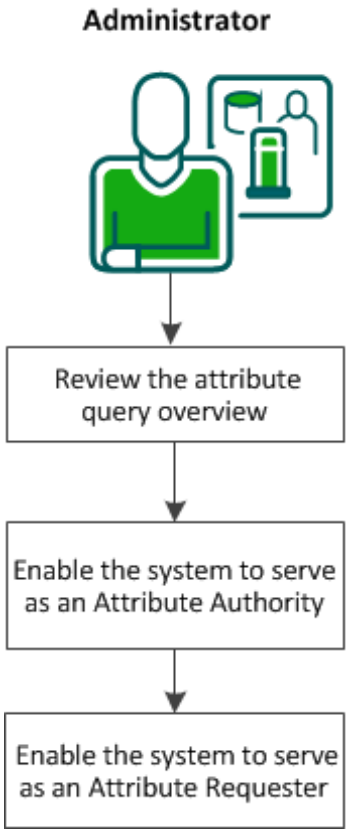
9. (Optional) Specify Partnership as the Protection Type in the Back Channel section. Select an authentication method. For more information about the back channel, click Help.
10. Save and activate the partnership.

The Identity Provider is now set up to serve as an Attribute Authority. This authority can now respond to attribute queries from a third-party SP.

Retrieve User Attribute Values from a Third-Party (SAML 2.0)

In a SAML 2.0 federated environment, a Service Provider sometimes requires information about a user that is not provided in the assertion. The Service Provider can request the values of predetermined user attributes. If the Identity Provider does not have these values, it can request the values from a third party. In a CA SiteMinder environment, this feature is referred to as a proxied attribute query.

The following diagram illustrates the process for enabling a proxied attribute query:



To enable a proxied attribute query, complete the following tasks:

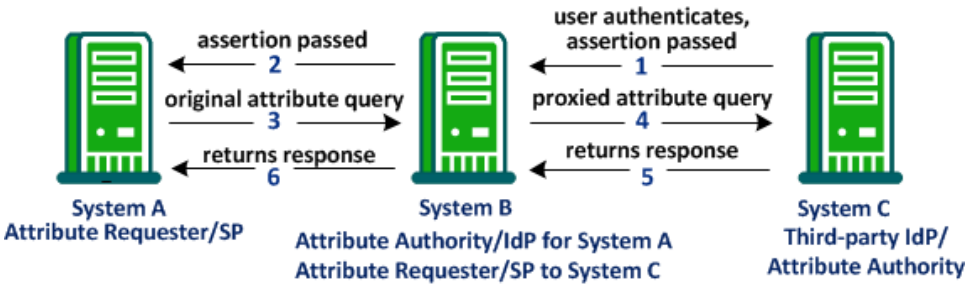
1. Review the proxied attribute query overview. (see page 120)
2. Enable the system to serve as an Attribute Authority (see page 121).
3. Enable the system to serve as an Attribute Requester (see page 122).

Proxied Attribute Query Overview

The proxied attribute query feature is based on the SAML 2.0 Assertion Query/Request profile and extends the search for user attributes. The Attribute Authority first searches the user directory and the session store for attributes. If the attribute is not found and the user initially authenticated at a third-party IdP, the request can be forwarded to the third-party IdP.

To implement a proxied attribute query, a single CA SiteMinder system acts as a relay point between two remote systems. To relay the request from one remote system to another, the single system takes on two roles. The system first serves as the Attribute Authority for the original Attribute Requester. The system also serves as an Attribute Requester to the third-party IdP. As the Attribute Requester, the system proxies the attribute query to the original IdP.

The following figure shows how a single system processes the proxied query:



The following steps explain the flow of a proxied attribute query:

1. The user initially authenticates at the System C, the third-party IdP. System C generates an assertion and passes it to System B.
2. System B sends the assertion to System A, completing the initial single sign-on transaction between Systems A, B, and C. This single sign-on transaction is necessary to process a proxied attribute query.

3. After System A receives the assertion, the system determines that it needs other attributes that are not in the assertion. As the Attribute Requester, System A sends an attribute query to its Attribute Authority/IdP, System B.
4. System B determines that System A requires attributes that are not in its user directory or session store. To retrieve the attributes, System B generates a new query request. It sends the new query to System C, the third-party IdP, where the user originally authenticated. This new query is the proxied query.
5. System C returns a response with the attributes to System B. System B saves the attributes in its session store.
6. System B, in its role as the Attribute Authority, returns its own response with the attributes to System A.

Important! The configured attribute names and the name format (unspecified, uri, or basic) at System A must match the names of these attributes at System C. This information is communicated before any transactions occurs.

Enable the System to Serve as an Attribute Authority (IdP->SP)

To implement a proxied query transaction, configure two partnerships on the same CA SiteMinder system:

- An IdP-to-SP partnership
- An SP-to-IdP partnership

For CA SiteMinder to serve as an Attribute Authority, modify an existing IdP-to-SP partnership or create a partnership. In this partnership, CA SiteMinder is the local IdP/Attribute Authority and the remote partner is the SP/Attribute Requester.

Note: This system also serves as the Attribute Requester in the SP-to-IdP partnership.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Partnerships.
3. Select the IdP-to-SP partnership that you want to modify or create a new one.
4. Navigate to the SSO and SLO step of the partnership wizard.
5. Select Enable in the Attribute Service section of the dialog.
6. Enter a number of seconds for the Validity Duration.
7. (Optional) Specify whether to require that the attribute query is signed, and the signing requirements for attribute assertions and responses.
8. Select Enable Proxied Query.

9. Enter the search specifications for the appropriate user directory name space in the User Lookup section. The Attribute Authority uses this search specification to disambiguate the user.

   An example for an LDAP user directory is uid=%s. At least one search specification is required.

10. (Optional) Specify Partnership as the Protection Type in the Back Channel section. Select an authentication method. For more information about the back channel, click Help.

11. Save and activate the partnership.

The system can now serve as an Attribute Authority to the original Attribute Requester.

Enable the System to Serve as an Attribute Requester (SP->IdP)

To implement a proxied query transaction, configure two partnerships on the same CA SiteMinder system:

- An IdP-to-SP partnership
- An SP-to-IdP partnership

Note: Partnership federation supports the SP as an Attribute Requester only for the proxied attribute query feature.

For CA SiteMinder to serve as an Attribute Requester, modify an existing SP-to-IdP partnership or create a partnership. In this partnership, CA SiteMinder is the local SP/Attribute Requester and the remote third party is the remote IdP/Attribute Authority.

Note: This system also serves as the Attribute Authority in the IdP-to-SP partnership.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Partnerships.
3. Select the SP-to-IdP partnership that you want to modify or create a new one.
4. Navigate to the SSO and SLO step of the partnership wizard.
5. Select Enable and Enable Proxied Query in the Attribute Requester Service section.
6. Provide a URL for the remote IdP in the Attribute Services section.
7. Provide the format, type, and value for the Name ID.

8. (Optional) Select an authentication type for the back channel. For information about the back channel, click Help.
9. Save and activate the partnership.

The Service Provider can now serve as an Attribute Requester.

User Consent at a SAML 2.0 IdP

A CA SiteMinder Identity Provider supports the user consent feature for SAML 2.0. User consent requires that the Identity Provider asks the user to grant permission before it sends an assertion to a partner. If you enable user consent at the Identity Provider, CA SiteMinder prompts the user for consent. The Identity Provider passes the consent value in an assertion.

The consent validity period is 5 minutes. When the Identity Provider redirects the user to the consent page, the user has 5 minutes to grant consent and be redirected back to the Identity Provider. The Identity Provider then generates the assertion and sends it to the Service Provider. These tasks must be complete in the 5-minute time period. If the time expires before the Identity Provider generates an assertion, it does not pass on the user identity.

Consent applies only to a single assertion. After the Identity Provider generates an assertion, it deletes all record of consent being granted. The same user can return to an Identity Provider before the 5-minute validity period expires, but the Identity Provider still prompts the user for consent.

Note: The validity period is not configurable.

Example

User1 logs in and authenticates at MyWorkPlace.com at 2:00PM. MyWorkPlace is acting as an Identity Provider. At 2:03PM, the user selects a link to the partner company that runs travel specials for employees. User1 is redirected to a form that asks for consent before sending User1 to ExampleTravel.com. User1 takes a phone call before completing the consent form. The time is now 2:10PM. MyWorkPlace does not generate an assertion because the validity period has expired.

If User1 grants consent promptly and is redirected back to the Identity Provider by 2:05PM, the Identity Provider generates an assertion. Only 2 minutes pass between consent and assertion generation, so the validity period is still active.

Configuring user consent requires that you:

- Enable user consent.
- Provide the name of a user consent form.

  The Identity Provider sends the custom form to the user to get consent.

If the Identity Provider includes a user consent attribute in the assertion response, only the following URI is used:

`urn:oasis:names:tc:SAML:2.0:consent:obtained`

User consent is also configurable at the Service Provider. A Service Provider can require the Identity Provider to pass the user consent value in the assertion response.

Customize a User Consent Form

CA SiteMinder ships with a *consent to federate* form named ca_defaultconsentform.html. The Identity Provider sends the custom form to the user to get consent. The default consent form is in the directory %NETE_WA_ROOT%\customization. %NETE_WA_ROOT% is the location of the Web Agent Option Pack.

You can write a custom form instead of using the default consent form and specifying the form in the Administrative UI.

Follow these steps:

1. Create the custom HTML form. Modify the form and replace values for the following settings:

   $$userconsent_spid$$

   Represents the SP ID configured in the partnership

   $$userconsent_idpid$$

   Represents the IDP ID configured in the partnership.

2. Place the form in the directory %NETE_WA_ROOT%\customization.

   NETE_WA_ROOT is the system environment variable. %NETE_WA_ROOT% is the location of the Web Agent Option Pack. If the Web Agent and Web Agent Option Pack are installed on the same system, they are installed in the same directory, for example, webagent\customization.

3. Log in to the Administrative UI.
4. Navigate to Federation, Partnership Federation, Partnerships.
5. Select the IdP->SP partnership you want to modify.
6. Navigate to the SSO and SLO step in the partnership wizard.

7. In the SSO section:
   a. Select the Enable User Consent check box.
   b. Specify the name of the custom form in the User Consent Post Form field.

   Note: The User Consent Service URL is specified by default. You cannot change this value.
8. Navigate to the Confirm step when your configuration is complete and click Finish.

Enhanced Client or Proxy Profile Overview (SAML 2.0)

The Enhanced Client or Proxy Profile (ECP) is an application for single sign-on. An enhanced client is a browser or some other user agent that supports the ECP functionality. An enhanced proxy is an HTTP proxy, such as a Wireless Access Protocol proxy for a wireless device.

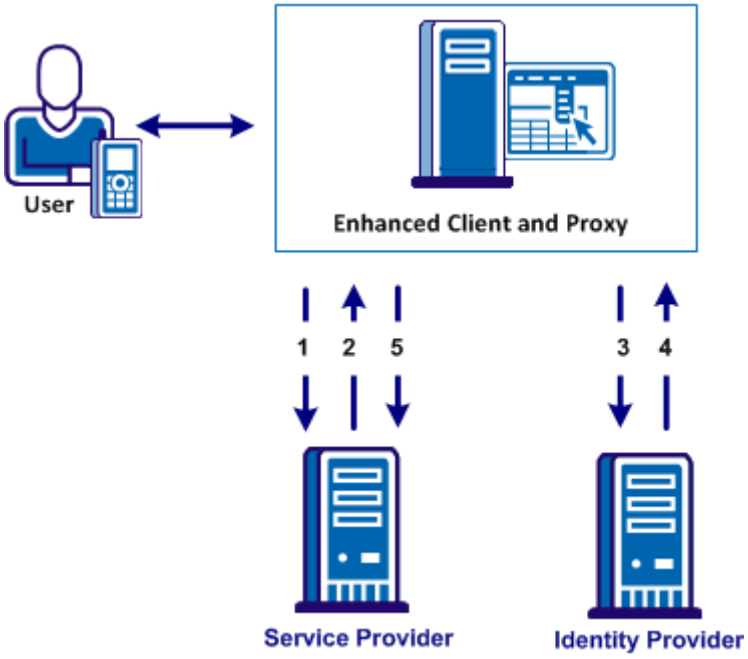
The ECP profile enables single sign-on when the Identity Provider and Service Provider cannot communicate directly. The ECP acts as the intermediary between the Service Provider and the Identity Provider.

In addition to acting as an intermediary, the ECP profile is useful in the following situations:

- For a Service Provider that expects to service enhanced clients or proxies that require this profile.
- When a proxy server is in use, such as a wireless access protocol (WAP) gateway in front of a mobile device with limited functionality.

You are responsible for obtaining or developing an ECP application. CA SiteMinder only processes the ECP requests and only responds to the ECP application in keeping with the SAML requirements.

The flow of the ECP profile is shown in the following illustration.



In an ECP communication, a user requests access to an application, for example, from a mobile phone. The application resides at the Service Provider and the identity information for the user resides at the Identity Provider. The Service Provider and Identity Provider do not communicate directly.

The flow of the call is as follows:

1. The ECP application forwards a reverse SOAP (PAOS) request to the Service Provider. The Identity Provider is not directly accessible by the Service Provider.

   The ECP entity is always directory accessible, unlike the Identity Provider.
2. The Service Provider sends an AuthnRequest back to the ECP application.
3. The ECP application processes and modifies the AuthnRequest and sends it on to the Identity Provider.
4. The Identity Provider processes the request and returns a SOAP response to the ECP application. This response includes the assertion.
5. The ECP application passes a signed PAOS response back to the Service Provider.

Single sign-on proceeds and the user gains access to the application.

Configure ECP at the Identity Provider

To configure ECP, enable the feature at the Identity Provider and the Service Provider. The following procedure is for a CA SiteMinder Identity Provider.

Follow these steps:

1. Log in to the Administrative UI.
2. Select the local Identity Provider partnership that you want to modify.
3. Navigate to the SSO and SLO step in the partnership wizard.
4. In the SSO section, select the Enable Enhanced Client or Proxy Profile check box.
5. Navigate to the Confirm step and click Finish to save changes.

The Identity Provider can now process ECP calls.

Note: A single Service Provider object can handle artifact, POST, SOAP, and PAOS bindings for single sign-on requests. SOAP and PAOS are the bindings for the ECP profile. The Identity Provider and Service Provider determine the binding being used based on the parameters in a request.

Configure ECP at the Service Provider

To configure ECP, you must enable the feature at the Identity Provider and the Service Provider. The following procedure is for a Service Provider.

Follow these steps:

1. Direct the requests for a protected resource to the AuthnRequest service at the Service Provider. The following URL shows an example:

   https://*host*:*port*/affwebservices/public/saml2authnrequest
2. Log in to the Administrative UI.
3. Modify the relevant local Service Provider partnership.
4. Navigate to the SSO and SLO step in the partnership wizard.
5. In the SSO section, select the Enable Enhanced Client or Proxy Profile check box.
6. Navigate to the Confirm step and click Finish to save the change.

The Service Provider can now process ECP calls.

Note: A single Service Provider object can handle artifact, POST, SOAP, and PAOS bindings for single sign-on requests. SOAP and PAOS are the bindings for the ECP profile. The Identity Provider and Service Provider determine the binding being used based on the parameters in a request.

IDP Discovery Profile (SAML 2.0)

The Identity Provider Discovery (IPD) profile provides a common discovery service that enables a Service Provider to select a unique IdP for authentication. A prior business agreement between partners is established so that all sites in the network interact with the Identity Provider Discovery service.

This profile is useful in federated networks that have more than one partner providing assertions. A Service Provider can determine which Identity Provider it sends authentication requests for a particular user.

The IdP Discovery profile is implemented using a cookie domain that is common to the two federated partners. A cookie in the agreed upon domain contains the list of IdPs that the user has visited.

IDP Discovery Configuration at the Identity Provider

You configure the IDP Discovery profile in the IDP Discovery section in the SSO and SLO dialog.

Note: Click Help for a description of fields, controls, and their respective requirements.

Follow these steps:

1. Select the Enable IDP Discovery checkbox.
2. Set the value for the Service URL field to the Identity Provider Discovery Profile servlet. For CA SiteMinder, this URL is:

   http://*host:port*/affwebservices/public/saml2ipd

   host

   Represents the common domain that you specify in the Common Domain field.

   port

   Specifies the Apache HTTP or HTTPS port you specified when installing the product.

   The URL can also begin with https.
3. Specify the cookie domain in the Common Domain field.
4. (Optional) Select the Enable Persistent Cookie check box to preserve the common cookie in the browser.

IdP Discovery is enabled at the IdP.

IDP Discovery Configuration at the Service Provider

For the IDP Discovery profile, the Service Provider (SP) has to determine the Identity Provider (IdP) to which it sends authentication requests. The user that the SP wants to authenticate must have previously visited the Identity Provider and authenticated.

The SP has to redirect the user to its own IdP Discovery Service to retrieve the common domain cookie. The cookie contains the list of Identity Providers that the user has already visited. From this list, the cookie chooses the correct IdP and then sends an AuthnRequest to that IdP.

The IDP Discovery process is as follows:

1. The browser requests the site selection page at the SP.

   This site selection page is aware of the IDP Discovery Service URL.

2. The site selection page redirects the user to IDP Discovery Service URL, indicating that it wants to get the Common Domain Cookie.

3. The IDP Discovery Service gets the Common Domain Cookie, reads the cookie in its domain and redirects the user back to the site selection page. The discovery service provides Common Domain Cookie as a query parameter.

4. The SP populates the site selection page with IdP URLs at which the user has previously authenticated.

5. The user selects an IdP to perform the user authentication.

To configure IdP Discovery at the SP

1. Create a site selection page that requests the Common Domain Cookie from the IdP Discovery Service at the SP.

   CA SiteMinder comes with a sample site selection page, named IdpDiscovery.jsp that the SP can use to implement IdP Discovery. You can find the page in the following directory:

   web_agent_home/affwebservices/public

   The first link redirects the browser from one domain to the IdPDiscovery service in the common domain and retrieves the common domain cookie, named **_saml_idp**. When the IdP Discovery Service at the SP receives the request, the service obtains the common domain cookie and adds it as a query parameter. The IDP Discovery Service then redirects the user back to the IdPDiscovery.jsp site selection page in the regular domain. By default, the IdPDiscovery.jsp page displays only a list of IDs for the IdPs that it extracts from the common cookie. This list is static; there are no HTML links associated with the list that initiate communication with the associated IdP.

2. Edit the following link on the sample page for your SP site. The first part of the link specifies the common domain where the saml2idp cookie resides. The second part of the link specifies the regular domain where the IdPDiscovery.jsp resides.

   For example:

   ```
   <a href="http://myspsystem.commondomain.com/affwebservices/public
   /saml2ipd/?IPDTarget=/http://myspsystem.spdomain.com/affwebservices
   /public/IdpDiscovery.jsp&SAMLRequest=getIPDCookie">
   Retrieve idp discovery cookie from IPD Service</a>
   ```

   When the user is redirected back to the regular domain with the target site selection page, it now has the common cookie.

3. (Optional) Edit the IdPDiscovery.jsp site selection page so it displays an HTML link for each IdP. Each link triggers an AuthNRequest to the IdP to initiate single sign-on. By default, the IdPDiscovery.jsp page only displays a list of IDs for the IdPs that it extracts from the common cookie.

4. Use the edited site selection page to test IdP Discovery.

With IdP Discovery working, you can see the site selection page with a list of IdPs from which to select.

Single Sign-on to Office 365

CA SiteMinder® Federation enables single sign-on between enterprise users and Office 365 services. Federating to Office 365 removes the burden of hosting services locally. For example, an enterprise user logs in to the desktop email client but is unaware that the service is in the cloud. The sign-in experience with Office 365 is the same as if they were connected to an on-premise application.

The following profiles are available for single sign-on to Office 365:

WS-Federation Passive Requestor Profile

WS-Federation Passive Requestor Profile works with passive requestors, primarily web browsers, or browser-based applications that support HTTP. The passive profile enables single sign-on between these clients and Microsoft Office 365.

WS-Federation Active Requestor Profile

The Security Token Service (STS) implements the WS-Federation Active Requestor Profile. This profile enables single sign-on between SOAP-enabled desktop clients and the following Office 365 services:

- Exchange Online (Outlook)
- Lync Online
- Dynamics CRM Online

The clients send and receive SOAP messages using HTTP-POST requests and responses. Users can sign in with their enterprise credentials and gain access to Outlook and Lync.

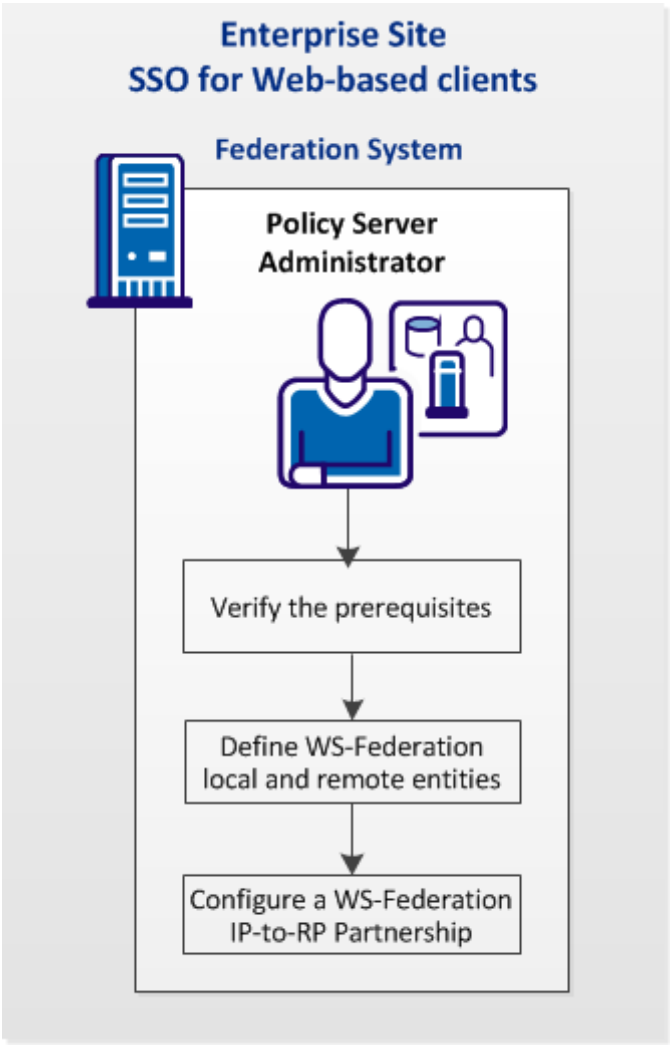
CA SiteMinder® Federation provides the STS service, which serves as an Identity Provider that Office 365 trusts. The STS service issues security tokens that Office 365 services can consume.

To implement single sign-on to Office 365, both WS-Federation profiles require a WS-Federation IP-to-RP partnership that is configured at the federation system.

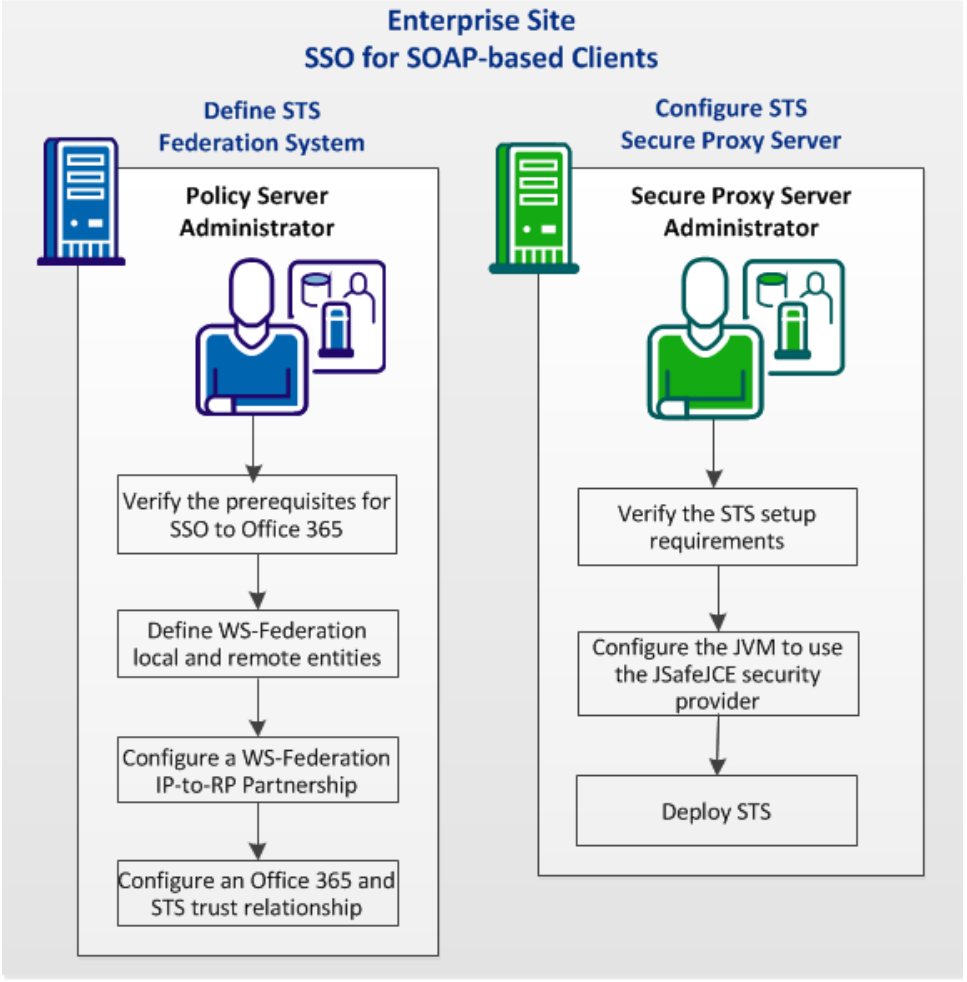
For the WS-Federation Active Requestor Profile, these other components are also required:

- The STS service that is enabled at the partnership
- STS configured on the CA SiteMinder Secure Proxy Server (SPS).

The following graphic shows the required configuration steps for web-based client SSO (Passive Requestor Profile):



The following graphic shows the required configuration steps for SOAP-based client SSO (Active Requestor Profile):



Complete the following tasks at the federation system:

1. Verify the prerequisites for SSO to Office 365 (see page 134).
2. Define WS-Federation local IP (see page 136) and remote RP (see page 138) entities.
3. Configure the WS-Federation IP-to-RP partnership (see page 140).
4. Configure a trust relationship between Office 365 and STS. (see page 141) (SOAP-based SSO only)

Complete the following tasks on CA SiteMinder® SPS for deploying STS (SOAP-based SSO only):

1. Verify the STS setup prerequisites (see page 143).
2. Configure the JVM to use the JSafeJCE security provider (see page 144).
3. Deploy STS on CA SiteMinder® SPS.

For supplemental configuration details about single sign-on to Office 365, view the appropriate runbook in the CA SiteMinder Federation Cloud Runbook Library. You need a login to view this content.

Verify the Prerequisites for SSO to Office 365

For single sign-on to Office 365, be aware of the requirements for:

- Office 365 setup
- CA SiteMinder user directory

Office 365 Setup Requirements

- Register and obtain an Office 365 domain. The plan that you register for must support single sign-on.
- Register a domain that you own.
- Add your domain to the Office 365 domain.
- Update the DNS record for the Office 365 domain that you own.

For information on configuring your deployment to work with Office 365, refer to the relevant Microsoft documentation for instructions on how to:

- Register and subscribe to Office 365.
- Configure directory synchronization between the on-premise and Office 365 user directory.
- Establish a trust relationship with Office 365 and the on-premise Secure Proxy Server that is configured with STS.

For supplemental configuration details about single sign-on to Office 365, view the appropriate runbook in the CA SiteMinder Federation Cloud Runbook Library. You need a login to view this content.

CA SiteMinder User Directory Requirements

- When you configure a user directory connection in the Administrative UI, verify that the ImmutableID and the UPN attributes exist for federation users. The values for these attributes in the on-premise user directory must match what is in the Office 365 directory.

  The Immutable ID and the UPN are required. Supply these values when you configure the WS-Federation partnership.

More information:

Verify the STS Setup Requirements (see page 143)

Configure a WS-Federation Partnership with Office 365

Configure a WS-Federation partnership with Office 365. A WS-Federation IP-to-RP partnership is necessary for either web-based or SOAP-based client SSO.

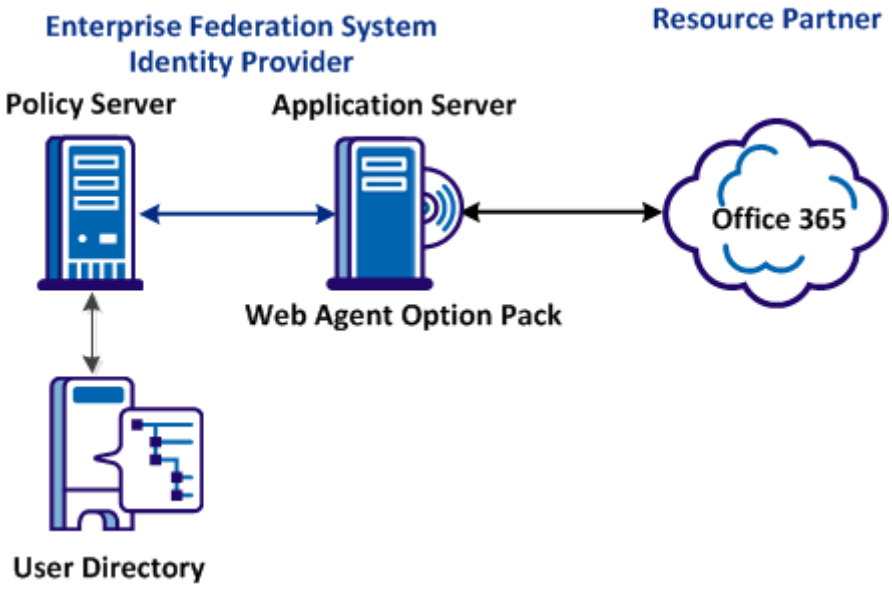
In this partnership:

- CA SiteMinder is the Identity Provider (IP)
- Office 365 is the Resource Partner (RP)

The differences between this partnership to configure WS-Federation Passive Requestor Profile and Active Requestor Profile are:

- Enabling STS for the Active Requestor Profile
- Configuring Sign-out, which is optional. Sign-out is only relevant for WS-Federation Passive Requestor Profile.

The following graphic shows a recommended deployment for this federated solution.



Define a Local IP Entity for an Office 365 Partnership

The on-premise federation system is the Identity Provider in the partnership with Office 365. As the Identity Provider, the system issues the security token containing the SAML 1.1 assertion.

Create a local Identity Provider entity with a SAML 1.1 token.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Entities.
3. Click Create Entity.

   The Create Entity dialog displays.
4. Select **Local** to indicate that you are creating an entity that is local to your site.
5. Configure the remaining fields:

   New Entity Type

   Select WSFED Identity Provider.

   SAML Token Type

   SAML 1.1

6. Click Next to configure specifics about the entity.

   In the Configure Entity step, complete all required fields in the dialog. Pay particular attention to the following fields:

   Entity ID

   Enter the IssuerURI specified in the Office 365 domain.

   For this local partner, the Entity ID does not have to be unique.

   Entity Name

   Enter any name that identifies this local IP. The Entity Name identifies an entity object in the policy store and it must be a unique value. This value is for internal use only; the remote partner is not aware of this value.

   Base URL

   Specify the URL for the system. For communication with Office 365, this URL must be over an SSL connection. For example, https://fedserver.example.com.

   Disambiguation ID (required for Office 365)

   Set this ID only when there are multiple partnerships between the same IP and RP, and your company has separate business units with their own relationship with Office 365. Office 365 uses a single ID to identify itself as an RP. CA SiteMinder Federation does not allow multiple partnerships with the same IP or RP ID. A disambiguation ID enables the system to differentiate partnerships with a unique logical path suffix for the service URLs given to a specific partner. Only one federation service exists, but the suffix that is combined with the RP ID creates a unique partnership lookup key.

   Example: microsoftonline

   The Disambiguation ID is appended to federation service URLs so requests go to the correct remote partner.

   Example:

   Passive Requestor Service URL:
   https://fedserver1.forwardinc.com/affwebservices/public/wsfeddispatcher/microsoftonline

   "microsoftonline" is the disambiguation ID.

   Enter an alphanumeric string but do not use any special characters.

   Sign-Out Confirm URL

   Specifies the URL at the Identity Provider that performs sign-out.

   Default: http://*ip_server:port*/affwebservices/signoutconfirmurl.jsp

   ip_server:port

   Specifies the server and port number of the Identity Provider system. The system is hosting the Web Agent Option Pack or the SPS federation gateway, depending on which component is installed in your federation network.

Signing Private Key Alias

For signing and encryption features, import the appropriate key and certificate pairs in the certificate data store.

Important! The public certificate that is associated with this private key must be imported into the Office 365 federation domain.

Supported Name ID Formats and Attributes

Unspecified

UPN Attribute

The UPN is the user principal name.

Assertion Attribute

UPN

Namespace

http://schemas.xmlsoap.org/claims

Note: Microsoft provides this value. Enter the name space value as it is shown. Office 365 requires this exact value.

Immutable ID Attribute

The ImmutableID is a unique attribute that distinguishes the user in the on-premise Microsoft directory.

Assertion Attribute

ImmutableID

Namespace

http://schemas.microsoft.com/LIVEID/Federation/2008/05

Note: Microsoft provides this value. Enter the name space value as it is shown. Office 365 requires this exact value.

7. Click Confirm when all the required fields are complete.
8. Configure the remote entity.

Define a Remote RP Entity for an Office 365 Partnership

Create a remote Resource Partner that represents Office 365. To define the entity, import metadata, if it is available, or configure the entity using the following procedure.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Entities.

3. Click Create Entity.

   The Create Entity dialog displays.

   Note: Click Help for a description of fields, controls, and their respective requirements.

4. Select Remote to indicate that you are creating an entity that is remote to your location.

5. Configure the remaining fields:

   New Entity Type

   Select WSFED Resource Partner.

   SAML Token Type

   SAML 1.1

6. Click Next to configure specifics about the entity.

7. In the Configure Entity step, complete the following required fields:

   Entity ID

   urn.federation:MicrosoftOnline.

   Entity Name

   Enter any name that identifies the RP.

   Remote Security Token Consumer Service URL

   Specify the URL for Office 365 security token service. Obtain this URL from Microsoft. This URL must be over an SSL connection. For example, https://login.microsoftonline.com.

   Remote Signout URL (Passive Requestor Profile only)

   Specify the URL for Office 365 sign-out service. Obtain this URL from Microsoft. This URL must be over an SSL connection. For example, https://login.microsoftonline.com.

   Note: For Office 365, the URLs for the security token consumer service and the signout URL are the same.

   Supported Name ID Formats

   Unspecified

8. Click Confirm after reviewing the configuration.

Configure a WS-Federation Partnership with Office 365

After you create the local IP and remote RP entities, configure a WS-Federation partnership. Steps that are specific to one or the other WS-Federation profile is noted.

Follow these steps:

1. Log on to the Administrative UI.
2. Navigate to Federation, Partnership Federation, Partnerships.
3. From the Create Partnership pull-down menu, select WSFED IP->RP.

   A dialog opens with the partnership wizard displayed at the top.
4. In step 1 of the partnership wizard, complete the required fields for a standard federated configuration.
   a. For Active Requestor Profile only, select the **STS for WSFED Active Profile** check box.
5. Optionally, select Enable Metadata Exchange to create a federation metadata document that reflects the active profile endpoints and data. The URL for this document is:

   https://*sps_host*/affwebservices/public/FederationMetadata/*partnership_name*

   Note: *sps_host* is the Secure Proxy Server with STS configured.

   The metadata document provides details about the partnership, such as WS-Federation passive and active profile end points and data.
6. In step 2 of the partnership wizard, select federation users for this partnership.
7. In step 3 of the partnership wizard, enter the required settings for the Name ID:

   NameID Format

   Unspecified

   Name ID Type

   User Attribute

   Name ID Value

   Immutable ID assigned by Office 365
8. Remain at step 3 of the wizard and complete the Assertion Attribute settings:
   a. Confirm that the UPN and ImmutableID assertion attributes are inherited from the local IP entity (see page 136). If you did not add these attributes at the entity level, specify them here.
   b. Set the Type field to User Attribute for both attributes.
   c. Set the Value field to the user directory attribute that has the UPN and ImmutableID value respectively.

9. In step 4 of the partnership wizard, complete the following fields in the Authentication section:

   Authentication Mode

   Local

   Authentication URL

   If the WS-Federation Passive Requestor Profile is in use, complete this field. Ignore this field for the Active Requestor Profile.

   https://*web_agent_optionpack_system*/affwebservices/redirectjsp/redirect.jsp

10. Remain at step 4 of the wizard and complete the following fields in the SSO section and the SLO section:

    Audience

    urn:federation:MicrosoftOnline

    Security Token Consumer Service URL

    https://login.microsoftonline.com/

    Complete the Signout fields only for the WS-Federation Passive Requestor Profile. They are not relevant for the Active Requestor Profile.

    Signout Confirm URL (optional)

    Enter the URL for your deployment, assuming sign-out is configured.

    Signout URL (optional)

    https://login.microsoftonline.com/

11. Accept the defaults for the remaining settings, then advance to the next step.
12. In step 5 of the partnership wizard, enable the signature processing and select the alias for the proper private key/certificate pair.
13. Move to the Confirm step. Review the configuration and click Finish to save the partnership.

    You return to the Partnership List.

14. Select Action, Activate to activate the partnership.

STS is defined for your federated partnership. Now configure the STS component on the CA SiteMinder® SPS.

Configure a Trust Relationship between Office 365 and STS (SOAP-based SSO)

To enable SSO between SOAP-based clients and Office 365, configure a trust relationship between the Office 365 Sign-In Service and the on-premise server with STS. Set up this relationship after you purchase an Office 365 subscription and after you configure directory synchronization.

Configure the trust relationship using the Windows Powershell commands. The command that initially configures the trust relationship is Set-MsolDomainFederationSettings. Run this command at your enterprise. For the correct procedure, see the Microsoft documentation on Windows Powershell.

This command can take command arguments such as:

- Domain
- ActiveLogOnUri (the ws-username endpoint of the on-premises STS)
- PassiveLogOnUri
- IssuerUri
- MetadataExchangeUri
- SigningCertficate

These command arguments are enough to establish a trust relationship from the Office 365 to the on-premises Secure Proxy Server system with STS.

To determine the STS endpoints for the command arguments, view an existing WS-Federation partnership in the CA SiteMinder Administrative UI. These endpoints are based on the values in the WS-Federation IP-to-RP partnership. Use these end points when configuring Office 365 to trust the Secure Proxy Server system with STS.

Note: To see the configuration of a specific WS-Federation partnership, select Action, View next to that partnership.

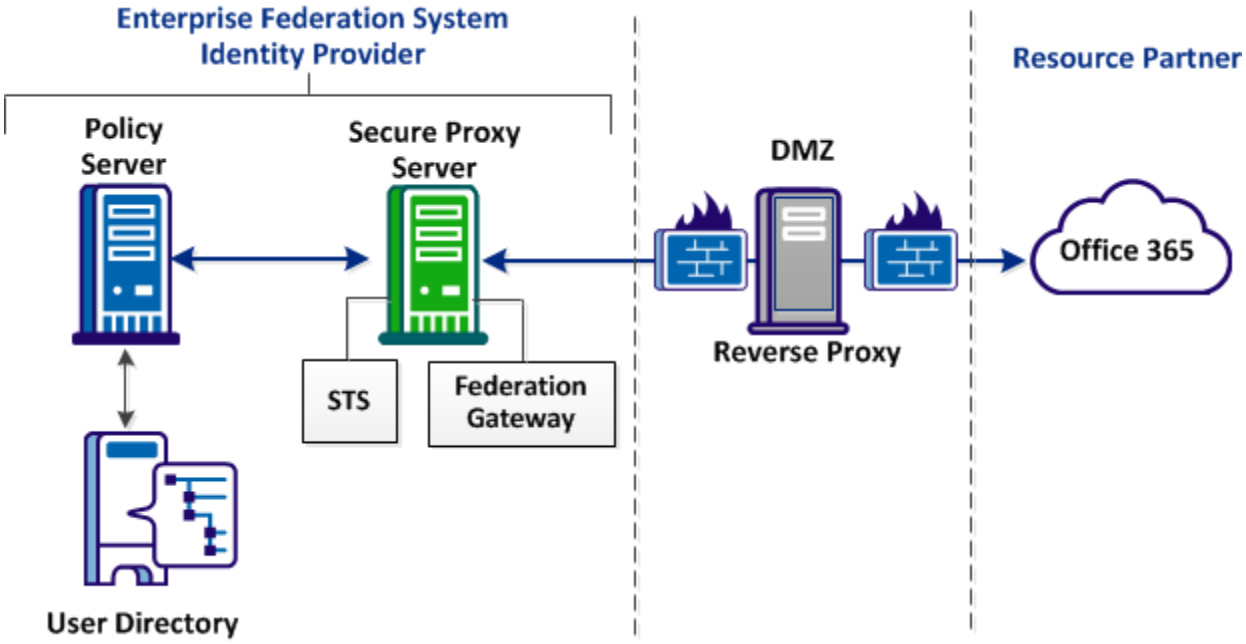
Configure CA SiteMinder® SPS

For CA SiteMinder to implement single sign-on using the WS-Federation Active Profile, an STS web service is needed. To manage requests from Office 365, deploy STS on CA SiteMinder® SPS.

Note: The STS web service must be hosted on CA SiteMinder® SPS at your enterprise. The service cannot be hosted on a different CA SiteMinder platform.

When a client application tries to connect to Office 365, Office 365 issues a token request to STS. STS issues a security token with a SAML 1.1 assertion that Office 365 can consume. The client application is able to access Office 365.

The following graphic shows a recommended deployment for this federated solution. Many ways to route traffic to the STS are available.



Follow these steps:

1. Verify the STS setup requirements. (see page 143)
2. Configure the JVM to use the JSafeJCE security provider (see page 144).
3. Deploy STS.

Verify the STS Setup Requirements

Before you deploy STS on CA SiteMinder® SPS, complete the following requirements:

- Install and configure CA SiteMinder® SPS.

  Note: Two or more secure proxy server systems behind a load balancer are recommended, but it is not required.

- Obtain the partnership name from the administrator of the CA SiteMinder® Federation system. Specify this name in the STS Context setting when you deploy STS.
- Enable SSL on CA SiteMinder® SPS.

- Verify that traffic originating from Office 365 can reach the on-premise CA SiteMinder® SPS with STS.
- The secure proxy server system with STS must be reachable by internal traffic and by external traffic from outside the enterprise firewall. Forwarding external traffic can require that you install a proxy in the DMZ. Configure the proxy so it can forward traffic from outside the firewall to the STS.

Configure the JVM to Use the JSafeJCE Security Provider

To enable encryption, configure the JVM that is running the CA SiteMinder® SPS so it uses the JSafeJCE Security Provider.

Follow these steps:

1. Download the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files package for the Java version you are using from the Oracle website.
2. Navigate to the following location:

   Windows

   JVM_HOME\lib\security

   UNIX

   JVM_HOME/lib/security

   JVM_HOME

   Defines the location where Java Runtime Environment (JRE) is installed in JDK of your installation.
3. Patch the following files with the files from the JCE Unlimited Strength Jurisdiction Policy Files package:
   - local_policy.jar
   - US_export_policy.jar
4. Open the java.security file.
5. Add the following line in the List of Providers section JSafeJCE is added as the second security provider:

   ```
   security.provider.2=com.rsa.jsafe.provider.JsafeJCE
   ```
6. Increment the order of preference of the other security providers by 1.
7. Add the following line at the end of the existing security providers list. This line sets the initial FIPS mode of JSafeJCE:

   ```
   com.rsa.cryptoj.fips140initialmode=NON_FIPS140_MODE
   ```
8. Save the changes.

The following example shows the List of Providers section of the java.security file after you configure the JVM:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.rsa.jsafe.provider.JsafeJCE
security.provider.3=sun.security.rsa.SunRsaSign
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC
security.provider.10=sun.security.mscapi.SunMSCAPI
com.rsa.cryptoj.fips140initialmode=NON_FIPS140_MODE
```

Deploy STS

To support the WS-Federation Active Requester Profile, deploy STS on CA SiteMinder® SPS.

Follow these steps:

1. Open the SPS Administrative UI.
2. Navigate to Web Services, Security Token Service.
3. Click Add.
4. Complete the following fields:

   STS Name

   Defines the name of the STS web service. Enter the partnership name that is defined in the Administrative UI.

   STS Context

   Defines the STS context path. Specify the name of the WS-Federation partnership that is defined in the Administrative UI. Enter the value using the syntax */partnership_name.*

   Example: /Office365Cloud

5. Click OK, and then click Save.
6. Restart the SPS for to apply the STS changes.
7. Log in to the SPS machine with root credentials.
8. Change to the "**/opt/CA/secure-proxy/proxy-engine/**" directory by using the following command:
   `#cd  /opt/CA/secure-proxy/proxy-engine/`
9. Stop the SPS by using following command:
   `#,/sps-ctl stop`
10. Start the SPS by using following command:
    `#,/sps-ctl startssl`
11. Test single sign-on to Office 365.

Note: When deploying STS instances in High Availability environments, deploy STS from each SPS leg. Specifically, the administrator must open the proxyui URL from each leg, and then deploy the same STS instance. This ensures STS is available from both SPS legs so that load balancer requests sent to any leg works correctly.

Test and Troubleshoot SSO to Office 365 (Active Requestor Profile)

Test SSO

Verify the WS-Federation configuration and the STS deployment by signing in to Lync or Outlook.

Follow these steps:

1. Log in to Lync or Outlook on the system in your enterprise.
2. Confirm that you get logged in and that you can use the application as if it were installed locally.

Troubleshooting SSO Issues

For WS-Federation partnerships and connectivity issues, use the following methods of investigation:

- Use the WS-Fed Passive Requester profile and verify single sign-on with Microsoft online.

  From a web browser, go to http://portal.microsoftonline.com or Microsoft exchange online. Try logging in with enterprise credentials. If you can successfully log in from the browser but not from the enterprise client, check the setup of the on-premise STS.

- Review what Office 365 knows about CA SiteMinder as a federation partner and what it knows about federation users. To examine the state of partnerships and users, run the following Microsoft Powershell commands:

  Get-MsolDomainFederationSettings

  Shows the information Microsoft has about your domain, that is, your enterprise. Review the settings and confirm whether they are accurate. Incorrect information can be a cause of federated communications problems.

  Get-MsolUser

  Shows the information Microsoft has about a particular user. Review the user settings and confirm whether they are accurate. Incorrect information can be a cause of federated communications problems.

- Validate the connectivity between your enterprise and Microsoft using Microsoft Remote Connectivity Analyzer. This tool lets you identify connectivity problems with Outlook, Lync, and Office 365. Locate this tool at https://www.testexchangeconnectivity.com/.

For any problems with the STS component, use the following logs and files:

- Review STS logs to verify that STS is operating, and whether there are authentication failures. Check the logs at *secure-proxy_install_dir*/proxy-engine/logs/*partnership_name*.log.

  Look for a message that says the STS initialization is complete. This message indicates that STS is running.

- Configure the log settings in the agent-log4j.xml configuration file. Set the log level for all categories to DEBUG so the system records the most detailed information in the *partnership_name*.log. The agent-log4j.xml file resides in the following directory:

  secure-proxy_install_dir/proxy-engine/conf/sts-config/*partnership_name*/config/

  Also, set the Checkpoint logger setting, <category name="com.ca.CheckPointLogger," to a priority value of "INFO." This setting writes checkpoint log messages for authentication activities and assertion generation. Checkpoint log messages are descriptive messages with codes that reflect the operation of the STS component.

  The section Federation Trace Logging (see page 255) describes checkpoint messages.

- Examine the WSDL file and confirm that the on-premise STS is responsive. Open a browser and go to http://*sts.company.com*/*partnership_name*?wsdl. The string *sts.company.com* is a place holder for the STS URL. You can find the STS URL in the WS-Federation partnership that is configured in the Administrative UI.

SAML 2.0 HTTP-POST Binding Configuration

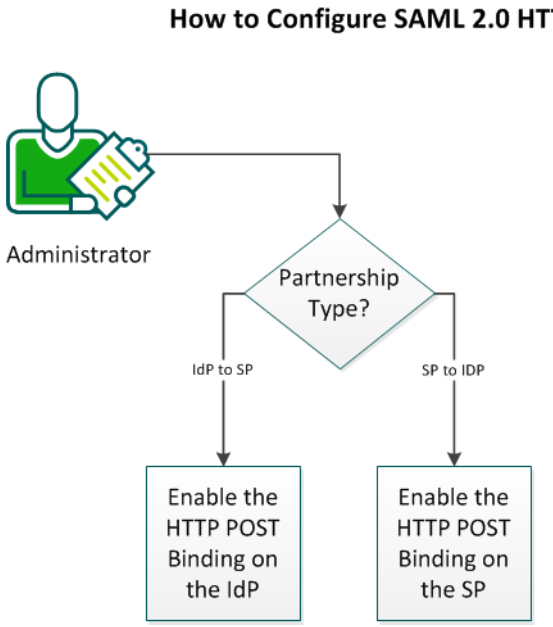
For single sign-on and single log-out requests, you can enable SAML 2.0 HTTP-POST binding as a method for exchanging requests and responses. The binding maps SAML protocols to standard messaging formats and communications protocols.

Note: The authentication request binding is different than the SSO binding. The SSO binding determines the profile that dictates how assertions, protocols, and bindings work together to handle a specific use case.

This procedure assumes that you are familiar with federated environments and have created and activated one or more of the following partnerships:

- IdP to SP
- SP to IdP

The following graphic describes how to enable SAML 2.0 HTTP POST binding:



Follow these steps:

1. Perform the appropriate task for your type of partnership:
   - Enable the HTTP POST binding at the IdP (see page 149).
   - Enable the HTTP POST binding at the SP (see page 150).

Enable the HTTP POST Binding at the IdP

You can enable the HTTP-POST binding at the IdP.

Important! Before you configure the authentication request binding, enable the session store. For the IdP to handle an authentication request that is delivered using HTTP-POST binding, the IdP must store the request in the session store.

Enable the Session Store

Follow these steps:

1. Open the Policy Server Management Console and select the Data tab.
2. Set the following fields

   Database

   Session Store

   Storage

   Select the storage repository.

   Session Store Enabled

   Check this box.
3. Complete the Datasource information.
4. Click OK to save the changes.

Configure the binding in the Administrative UI

Follow these steps:

1. Open the Administrative UI.
2. If the partnership that you want to modify is active, deactivate it.
3. Click Modify to open the partnership wizard.
4. Navigate to the SSO and SLO step.
5. In the SSO section, select HTTP-POST for the Authentication Request Binding.

   Note: You can select the HTTP-Redirect and HTTP-POST bindings together for authentication requests.
6. (Optional) In the SLO section, select the HTTP-POST check box.

   Note: You can select more than one SLO binding.

7. Specify a SLO service URL with a binding that matches the SLO binding. If you picked the HTTP-Redirect and HTTP-POST bindings, create two SLO service URLs, one for each SLO binding.
8. Complete any other partnership information as needed.
9. At the confirm step, click Finish.

HTTP-POST binding is now enabled.

Enable the HTTP POST Binding at the SP

You can enable the HTTP-POST binding for authentication and SLO requests at the SP.

Follow these steps:

1. Open the Administrative UI.
2. If the partnership that you want to modify is active, deactivate it.
3. Click Modify to open the partnership wizard.
4. Navigate to the SSO and SLO tab in the partnership wizard.
5. In the SSO section, select HTTP-POST for the Authentication Request Binding.

   Note: You can select the HTTP-Redirect and HTTP-POST bindings together for authentication requests.
6. Specify a remote SSO service URL with a binding that matches the Authentication Request Binding. For example, if you picked HTTP-Redirect and HTTP-POST bindings, create two SSO Service URLs, one for each binding.
7. (Optional) In the SLO section, select the HTTP-POST check box.

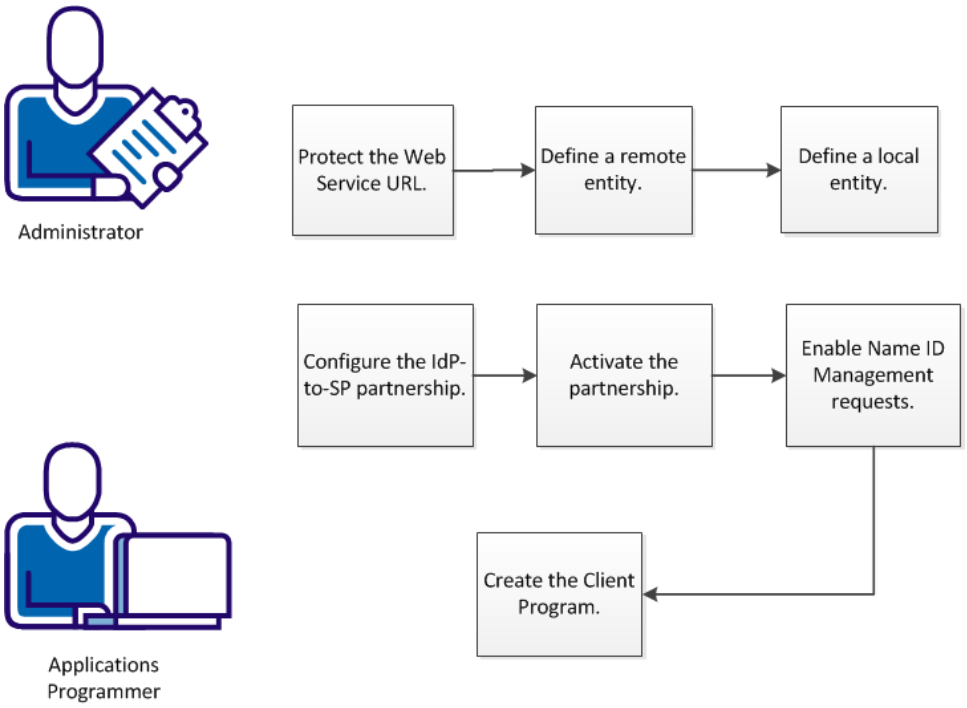
   Note: You can select more than one SLO binding.
8. Specify an SLO Service URL with a binding that matches the SLO binding. For example, if you picked HTTP-Redirect and HTTP-POST SLO bindings, create two SLO Service URLs, one for each binding.
9. Complete any other partnership information as needed.
10. At the confirm step, click Finish.

SSO HTTP-POST binding is enabled.

Configure the SAML 2.0 Name ID Management Profile

The SAML 2.0 Name Identifier profile lets you de-provision an individual user from a federated partnership. You can remove a user from a partnership for any number of reasons. For example, an employee can have left the company, or no longer requires SSO capability with a Service Provider. You make the de-provisioning request through a client application program.

The following diagram illustrates the process of implementing the SAML 2.0 Name Identifier profile:



To use the Name ID Management profile to de-provision a user requires these steps:

1. Protect the Name Identifier Management Administration Web Service URL. (see page 152)
2. Configure a remote entity for Name ID Management. (see page 152)
3. Create a local entity. (see page 153)
4. Configure a partnership for Name ID Management. (see page 153)
5. Activate the partnership. (see page 154)
6. Enable Name ID Management requests. (see page 154)
7. Create a client application to interact with the Name Identifier Web Service. (see page 155)

Protect the Name Identifier Management Administration Web Service URL

Customer applications can use the Name Identifier Management Administration Web Service to request the de-provisioning of a user from a partnership. This web service implements the REST interface.

The URL for this service is /affwebservices/saml2nidws. You protect this URL using CA SiteMinder Basic credentials. Include any users of this service in the user directories that are associated with the domain. CA SiteMinder policy administrators are not included by default. You can add them manually to an associated directory.

Configure a Remote Entity for Name ID Management

The first step in creating a partnership that supports Name ID management is to define the remote and local partners, or entities. You can configure the entity manually, or you can import the XML metadata. These following steps are for manual configuration.

Follow these steps:

1. Navigate to Federation, Partnership Federation, Entities in the Administrative UI.
2. Click Create Entity.
3. Select Remote (either IdP or SP, depending on your implementation).
4. Click Next to configure specifics about the entity.
5. Enter the values for Entity ID and Entity Name (required).
6. Click Add Row on the line for Manage Name ID Service URLs.
7. Select the SOAP binding. A remote entity can specify another binding, which is imported, but unused.
8. Enter the Location URL, which specifies the URL of the Name ID Management service. This value is as follows:

   http://*sp_server:port*/affwebservices/public/saml2nidsoap
9. Leave the Response Location URL field blank. The Response Location URL for the SOAP binding is the same as the Location URL.
10. Select any supported Name ID formats from this list.
11. Complete any other fields that your implementation requires.
12. Click Next to confirm the entity configuration.
13. Click Finish.

Create a Local Entity

The first step in creating a partnership that supports Name Id Management is to define the remote and local partners, or entities. You can configure the entity manually, or you can import the XML metadata. See Federation Entity Configuration (see page 61) for information if you are unfamiliar with creating an entity in a partnership.

Important! You can select any Name ID format for user de-provisioning or de-linking. Dynamic account linking supports only Persistent Identifier format. In cases where you are implementing account linking and de-linking, select the Persistent Identifier Name ID format.

Configure a Partnership for Name ID Management

Enabling the Name ID Management feature requires some configuration for a new partnership or existing partnership. Either the local or the remote entity can initiate a request to de-provision a user from the partnership.

Follow these steps:

1. Navigate to the SSO and SLO dialog.
2. Configure the Authentication and SSO sections if they are not configured already.
3. Navigate to the Manage Name ID section.
4. Select SOAP in the MNI field.

   This selection enables Name ID Management in your partnership. See the online help for descriptions of these options.
5. (Required) Specify a SOAP timeout value. This value is the number of seconds the runtime waits until the request to the remote provider times out.

   Default: 60 seconds.
6. (Required) Specify the Retry Count, which is the number of times a background request is attempted before declaring failure. The default is 3.
7. (Required) Specify the Retry Boundary, which is the number of minutes in the interval between retries. The default is 15 minutes.
8. Specify the Notification URL if you have selected the Enable Notification option. This URL is the location where to send an HTTP notification to the customer application. The notification includes the status of the de-provisioning request after it has completed:
   - Status 1 for successful de-provisioning
   - Status 0 for unsuccessful de-provisioning

9. Specify the Notify Timeout, which is the number of seconds after which the request is considered timed out.

   Default 60 seconds.

10. Specify the Notification Authentication Type (NoAuth or Basic). If you select Basic, provide a user name and password.

Note: Select Delete Name ID or the Enable Notification option or both in the MNI section for the feature to function properly.

These steps complete the Manage Name ID configuration.

Activate the Partnership

See Partnership Activation (see page 75) for details.

Enable Name ID Management Requests

A Web Agent Option Pack internal component named the Asynchronous Requesting Processor handles all requests to the Name ID Management Service. Only one Web Agent Option Pack can have this service running at one time. In addition to settings in the Administrative UI, you enable Name ID Management processing by specifying settings in the AffWebServices.properties file in the following locations:

- SPS: <SECURE_PROXY_HOME>/Tomcat/webapps/affwebservices/WEB-INF/classes
- WA+WAOP: <WEB_AGENT_HOME>/affwebservices/WEB-INF/classes

The AffWebServices.properties file contains the following settings that are related to Name ID Management:

ProcessBackgroundNameIDOperations

Specifies whether this system processes Name ID operations.

Default: False

Important! Set this value to True to enable Name ID Management for the Option Pack or SPS.

BackgroundProcessingInterval

Specifies the number of seconds between times the asynchronous processor check for a Name ID request. You can modify this value.

Default: 60 seconds

If you upgrade your Option Pack or SPS, the installer adds these settings with their default values to the new properties file.

Create a Client Application to Interact with the Name Identifier Web Service

The content of the client application is implementation-dependent. To request the removal of a user, use the Name Identifier Management Administration Web Service. The web service implements the following two HTTP methods:

- POST — to initiate a de-provisioning request.
- GET — to poll the status of a request.

The methods adhere to the OData protocol. Details about these methods follow.

Terminate Federation Membership

An administrator can terminate the federated membership for a user by using the following URL:

```
POST http://<server+port>/affwebservices/saml2nidws/terminate
```

This asynchronous request creates a ManageNameID event in XPS.

The POST body includes the following values:

UserDN

Disambiguates the user, because there is no SMSession. An example DN for LDAP is uid=user0001,ou=Engineering,o=security.com.

OperationType

Indicates a particular use case. The valid values are:

- sp – indicates an idp wishing to terminate a Federation with a specific Service Provider.
- idp – indicates a Service Provider wishing to terminate a Federation with a specific Identity Provider.

ProviderID

Determines which providers are part of the operation. For the sp and idp values, the ProviderID identifies a remote provider. If the OperationType is ‘sp’, the ProviderID represents a remote Service Provider object. If the OperationType is ‘idp’, the ProviderID represents a remote Identity Provider object.

The information in the POST body of the request is in JSON or AtomPub format. The following example is in JSON format:

```
{
"UserDN":"uid=user0001,ou=Engineering,o=security.com",
"OperationType":"sp",
"ProviderID":"http://company.example.com/SPID"
}
```

This request returns the resource that represents this persisted object, for example:

```
http://<server+port>/affwebservices/saml2nidws/terminate(<XID>)
```

<XID> is the XPS XID of the created object. The client can use this URL to poll for changes to this object.

The request can also be in full AtomPub format as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
<title type="text"></title><author><name></name></author>
<category term="NameidProducer.terminate"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"></categor
y>
<content type="application/xml">
<m:properties>
<d:UserDN>uid=user0001,ou=Engineering,o=security.com</d:UserDN><d:ProviderID>ht
tp://company.example.com/SPID</d:ProviderID>
<d:OperationType>SP</d:OperationType>
</m:properties>
</content>
</entry>
```

The POST service sets the following HTTP return codes:

HTTP Status	Description
201	Resource created
400	Bad request
415	Unsupported media type
500	Internal server error

Poll for Status

An administrator can use this service to request status of the asynchronous request using the following URL:

```
GET http://<server+port>/affwebservices/saml2nidws/terminate(<XID>)
```

The URL used to poll for the resource status.

The response returns the status of the request, either PENDING, COMPLETED, or FAILED.

Important! Before you make this request, be sure that the CssChecking parameter in the Agent Configuration Object is set to NO. This setting avoids a potential conflict in syntax between OData and a cross-site scripting attack.

The GET service sets the following HTTP return codes:

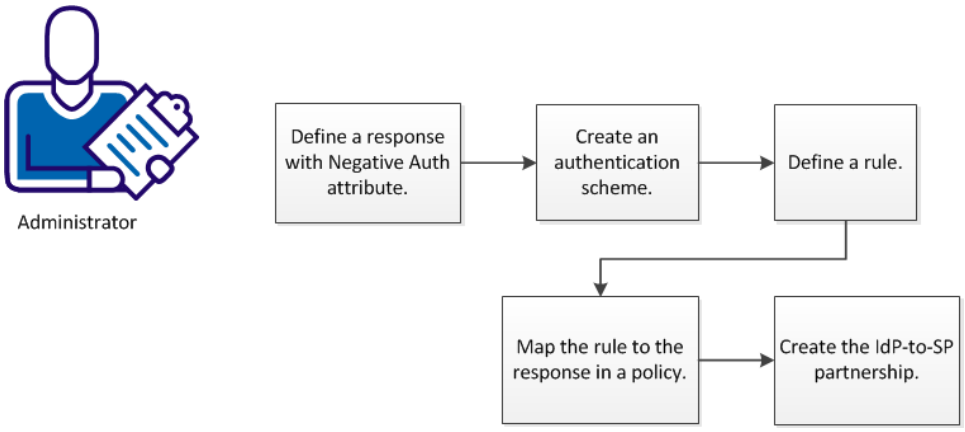
HTTP Status	Description
200	OK
400	Bad request
403	Forbidden (when CSS checking is on for the web agent)
415	Unsupported media type
500	Internal server error

Configure a SAML 2.0 Response for Authentication Failure

You can use this process to configure a non-assertion response to the Service Provider on authentication failure. When a SAML 2.0 authentication request is successful, the response to the Service Provider carries with it the authentication assertion. A rejected authentication request previously only resulted in the end user getting an error message. The Service Provider got no notification of the failed status. Because control returns to the Service Provider, the Service Provider can determine whether to redirect the user, or take any other appropriate action.

Important! For this feature to work, the Policy Server, the Web Agent, and the Web Agent Option Pack are all required to be at SM r12.52 or later.

The following diagram depicts the steps required to configure this functionality:



The process of configuring a response to the Service Provider on authentication failure includes the following procedures:

1. Define a response specifying the Negative Authentication Response attribute. (see page 158)
2. Create a Basic or Forms authentication scheme (see page 159).
3. Define a rule specifying the OnAuthReject action (see page 160).
4. Map this rule to the previously defined response in a policy (see page 161).
5. Configure an IdP-to-SP partnership to enable negative authentication response (see page 161).

Define a Response Specifying the Negative Authentication Response Attribute

Begin by defining a response using the WebAgent-OnReject-eGovNegResponse attribute type. Defining a response presupposes a defined domain.

Follow these steps:

1. Navigate to Policy, Domain, Responses.
2. Click Create a Response.
3. Select an appropriate domain, or create a new one.
4. Click Next.
5. Enter a name and description (optional) for this response in the General section.
6. Select the appropriate agent type, usually a SiteMinder web agent.
7. Click Create Response Attribute in the Attribute List section.
8. Select WebAgent-OnReject-eGovNegResponse from the drop-down list in the Attribute Type section.

9. Select Use Relative Target or enter a web server name in the Attribute Fields section.
10. (Optional) Select Use SSL Connection.

    Note: The selections that you make in this section are the basis for the script that is displayed in the pane in the Advanced section. See the online help for more information.
11. Select Cache Value Recalculate Value in the Attribute Caching section.
12. Click Ok to return to the Create Response: Define Response dialog.
13. Click Finish.

You have defined a response with the appropriate attribute to generate a response to the SP when an authentication fails.

Configure a Basic or Forms Authentication Scheme

You can configure a Basic or Forms scheme to generate a response on authentication failure to the SP.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK.
5. Enter a name and protection level.
6. Select Basic or Forms Template from the Authentication Scheme Type list.
7. Click Submit.

   The authentication scheme is saved and can now be assigned to a realm.

Configure a Rule for Authentication Event Actions

You can configure a rule to control actions that occur when users attempt to gain access to a resource. For a full SAML 2.0 response on authentication failure, select the OnAuthReject action.

The realm must be able to process authentication events. Verify that the Process Authentication Events option is selected. For information about how to create a realm, see the next topic.

Follow these steps:

1. Click Policies, Domain, Rules.
2. Click Create Rule.
3. Select a domain from the list, and click Next.
4. Select the realm that includes the resources that you want the rule to protect, and click Next.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.
5. Type the name and a description of the rule.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
6. Select Authentication events.

   The Action List populates with authentication events.

   Note: The Resource field is disabled because an authentication event applies to the entire realm. The Allow Access and Deny Access options are also disabled as they do not apply to authentication events.
7. Select the OnAuthReject action.
8. (Optional) In the Advanced section, set time restrictions and or active rule settings.
9. Click Finish.

   The rule is saved and applied to the specified realm and resource.

Map the Rule Using the OnAuthReject Actiton to the Appropriate Response

Associate the rule you created using the OnAuthReject action with the eGovNegResponse attribute in a policy.

Follow these steps:

1. Navigate to Policies, Domain Policies.
2. Select a policy.
3. Navigate to Rules
4. Verify that the rule you created with the OnAuthReject action is is in the list of rules.
5. Click Add responses next to that rule.
6. Select the response in you specified with the eGovNegResponse attribute type.
7. Save and exit.

You have associated your rule with the appropriate response.

Configure an IdP-to-SP Partnership to Support Negative Authentication Response

You enable a negative authentication response in the SSO configuration step of the IdP-to-SP partnership configuration.Select the Enable Negative Authentication Response check box.

See Single-Sign-on Configuration (see page 107) for further information.

Chapter 12: Configure Social Sign-on

You can configure CA SiteMinder® Federation (the federation system) to let users sign-on to a federated resource with their social networking credentials instead of the federation system credentials.

The social sign-on feature consists of the following features:

- Authentication of users using an OAuth authorization server such as Facebook so that users can sign-on to a federated resource using their OAuth authorization server credentials.
- Configuration of a credential selector page that provides users with various identity providers such as SAML 2.0 or Facebook as authentication choices. Users can choose an identity provider for authorization to sign-on to a federated resource.

The features are independent of each other and you can configure the federation system to implement either or both the features.

To use an external IdP using the OAuth protocol in a CA CloudMinder environment, you need to perform configuration procedures in the following environments:

- The SMPS Environment, including authenticating users using an OAuth Authorization Server
- The Management Console Environment
- The Tenant Environment

This section contains the following topics:

Chapter 13: Configure the SMPS Environment: Authenticate Users Using an OAuth Authorization Server

To configure the SMPS environment, you must authenticate users using an OAuth authorization server. To do this, configure single sign-on between the federation system and the OAuth authorization server.

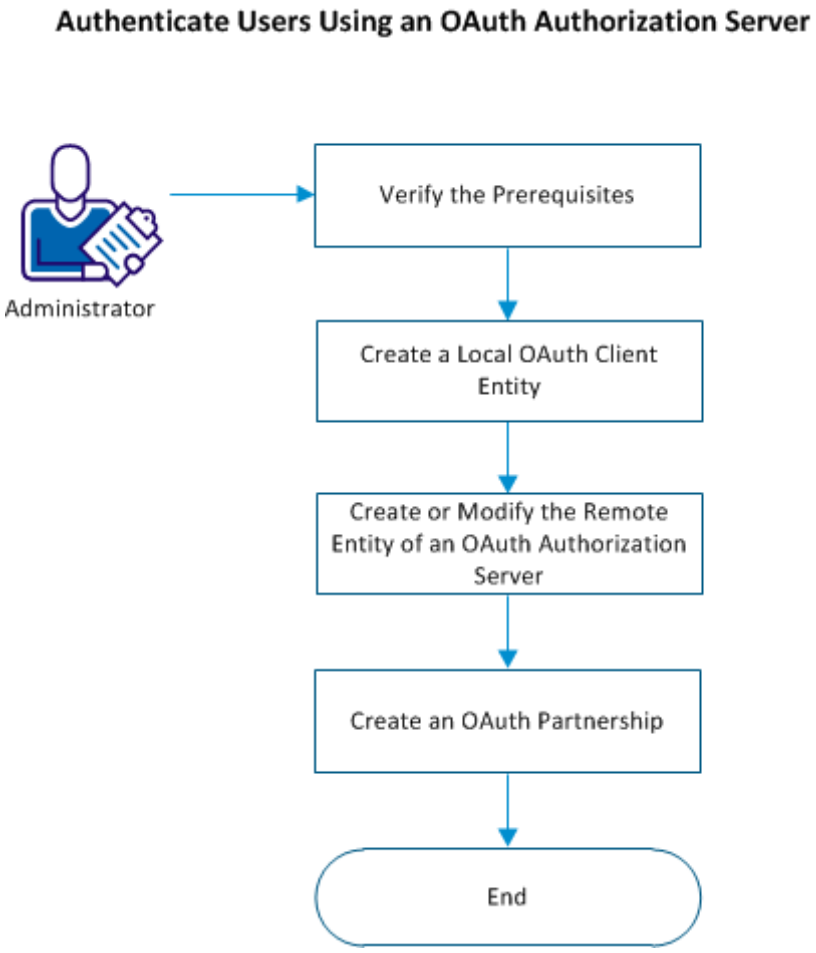
The federation system provides default support for the following OAuth authorization servers:

- **OAuth 1.0a**
- **OAuth 2.0**

The following process describes how the federation system processes a user request to access a federated resource:

1. The federation system redirects the user request to the OAuth authorization server specified in the user request.
2. The OAuth authorization server authenticates the user and sends an authentication response with claims about the user to the federation system.
3. The federation system verifies the authentication response, completes the authentication process, and authorizes the user to access the federated resource.

The following flowchart describes how you can authenticate users using an OAuth authorization server:



This section contains the following topics:

Verify the Prerequisites

Perform the following steps before you configure a partnership to configure single sign-on between the federation system and an OAuth authorization server:

- Enable SSL on the federation system.
- To use an OAuth authorization server that the federation system supports by default, perform the following steps before you invoke the partnership:
  - On a CA CloudMinder deployment, verify that the default CA certificate of the OAuth authorization server was imported.
  - Import any other existing CA signed certificate that is required using the smkey tool provided.
- To use an OAuth authorization server that the federation system does not support by default, obtain and import the SSL CA certificate of the OAuth authorization server before you invoke the partnership.
- Clear the policy server cache after importing the certificates.

Create a Local OAuth Client Entity

Create a local OAuth client entity for the partnership between the federation system and an OAuth authorization server.

Follow these steps:

1. Navigate to Federation, Entities, and click Create Entity.
2. Choose Local in Entity Location.
3. Select OAuth Client from New Entity Type.
4. Select the OAuth version, and click Next.
5. Enter the required values, and click Next.
6. Confirm the entered values and click Finish.

   The Redirect URL is constructed. Use this URL for initiating an OAuth transaction.

Create or Modify the Remote Entity of an Authorization Server

The system provides remote entities for each of the following OAuth authorization servers that are supported by default:

- **OAuth 1.0a**
- **OAuth 2.0**

The values of each remote entity are pre-configured with known values of the entity. You can modify the values to suit your federation environment or create a remote entity for any OAuth authorization server.

Follow these steps:

1. Perform *one* of the following tasks:

   Create a new remote entity:

   a. Navigate to Federation, Entities, Create Entity.

   b. Select Remote as Entity Location, and select OAuth Authz Server as the New Entity Type.

   c. Click Next.

   d. Enter the values and click Next.

   Modify the pre-populated values of a remote entity:

   a. Navigate to Federation, Entities, and search for the entity that you want to modify.

   b. Click the Actions option of the entity, and click Modify.

   c. Click Next to go to the Configure Entity tab.

   d. Modify the values and click Next.

2. Confirm the changes and click Finish.

Create an OAuth Partnership for Single Sign-On

To let the federation system retrieve user information from the authorization server, create an OAuth partnership between the OAuth authorization server as the asserting party and the federation system as the relying party.

Follow these steps:

1. Navigate to Federation, Partnerships and click Create Partnership.
2. Select the OAuth Client - Authz Server partnership type.
3. Configure the partnership information.
4. Confirm the values and click Finish.

An OAuth partnership is configured to let users sign-on to a federated resource using the OAuth authorization server credentials.

When the federation system receives a user request in the following format, the request is processed per the partnership configuration:

https://*baseURL_of_the_partnership*/affwebservices/public/oauthtokenconsumer?AuthzServerID=*authorization_server_id*

Or

https://*baseURL_of_the_partnership*/affwebservices/public/oauthtokenconsumer/*disambiguation_id*?AuthzServerID=<*authorization_server_id*>

The federation system is configured to implement the social sign-on feature.

Note: The Authorization URL constructed above should be configured as part of the External Authentication Scheme created for the tenant. You must then map it to an application in the CA CloudMinder tenant portal.

Migrate an OAuth Authentication Scheme Set-up to OAuth Partnership

If you configured an OAuth authentication scheme in your environment to authenticate users using an OAuth provider, you can migrate your authentication scheme set-up to a federation partnership.

Follow these steps:

1. Create a partnership between the OAuth client and the OAuth authorization server.
2. Perform *one* of the following steps:
   - If you want to use both the OAuth authentication scheme and an OAuth partnership simultaneously, update the existing redirect URL at the OAuth authorization server to the appropriate partnership redirect URL of the following format :
     `https://server:port/affwebservices/public/oauthtokenconsumer`
   - If you want to use an OAuth partnership instead of the OAuth authentication scheme,
     a. update the existing redirect URL at the OAuth authorization server to the appropriate partnership redirect URL of the following format:
     `https://server:port/affwebservices/public/oauthtokenconsumer`

     and

     b. Disassociate the realm created for authmethod in Siteminder.

     c. Modify the Authentication Method in Tenant Console that is currently set to:

     Either:
     `/chs/redirect/<tenantName>/<uniqueName>`

     Or:
     `/affwebservices/<tenantName>/<uniqueName.jsp>`

     To:
     `https://server:port/affwebservices/public/oauthtokenconsumer?AuthzServerID=<AuthorizationServerID>`

Configure the Management Console Environment

The topic shows you how to configure the attributes in the “openformatcookie” used by the user provisioning service to provision new users.

F**oll**ow these steps:

1. Log in to the Manage console.
2. Click Environments.
3. Click the Tenant name.
4. Click Advanced Settings, Miscellaneous.
5. Add the following five properties shown in the table below, and then Save the configuration.

Property	Value	Description
openformat.cookie.domain	ca.com	Enter the domain name with which the cookie needs to be created.
openformat.cookie.zone	SM	
openformat.cookie.name	DEFAULT	
openformat.cookie.encryption.password	Firewall	Password used User provisioning section in the partnership
openformat.cookie.encryptiontype	AES256/CBC/PKCS5Padding	Encryption algorithm used to generate the above password. AES128/CBC/PKCS5Padding AES192/CBC/PKCS5Padding AES256/CBC/PKCS5Padding 3DES_EDE/CBC/PKCS5Padding

Chapter 14: Configure the Tenant Environment

This section contains the following topics:

Create a CHS Application and Map the SSO Authentication Method

This topics shows you how to create a credential handling service (CHS) application, as well as how to map the social sign on authentication method.

Follow these steps:

1. Log in to the Tenant Console as CSP Administrator.
2. Navigate to Applications, Modify Application.
3. Search for the desired Application.
4. Select your <tenant_name>, and then click Select.
5. Click Add, and the desire application.
6. Click Submit.
7. Navigate to Applications, Authentication Methods, Modify Authentication Scheme.
8. Select the Authentication Method of your application to modify.
9. Select the Enabled checkbox.
10. From the Authentication Method Scheme drop-down list, select your application.
11. Update the Authentication URL :
    `https://<baseURL_of_the_partnership>/affwebservices/public/oauthtokenconsumer?AuthzServerID=<authorization_server_id>`

Or

`https://<baseURL_of_the_partnership>/affwebservices/public/oauthtokenconsumer/<disambiguation id>?AuthzServerID=<authorization_server_id>`

Enable OAUTH Self Registration

Eenable Self Registration for your application's OAUTH authentication.

Follow these steps:

1. Under the Tenant domain of your applications's OAUTH Realm, create a new Rule such as (OAuth_<*Your Application Name*>_SelfReg
2. Create a new Response (OAuth_SelfReg_Response) under the Tenant domain.

   Note: Ignore this step if SelfReg Response is already created for any other OAUTH Self registration.
3. Add the new Rule and Response to Tenant domain policy.

Enable the Self Registration Check Box

Log in to the Tenant console and select the Self Registration check box for the OAUTH Authentication method.

Follow these steps:

1. Login into Tenant console.
2. Navigate to Applications, Authentication Methods, Modify Authentication Method.
3. Select your Authentication Method.
4. Makes sure to select the "Enabled for Self Registration" check box.

Important! When user gets authenticated from a social sign-on page such as Facebook, the cspadmin must protect the Credential Handling Service (CHS) application with a Forms authentication scheme.

Specifically, if CloudMinder is acting as an OAuth Authz Server hub, and a user gets authenticated from a social sign-on page so that SMSESSION passed to L7 for validation, protect the resource with Forms authentication using the following format:
`/chs/redirect/<tenant name>/CHS App name_Used in L7/`

For example:
`/chs/redirect/layer7/Layer7IDP`

Troubleshooting Configure Social Sign-on

SSO Successful but Unable to Find a User

If the SSO is successful, but unable to find a user, check if the filed in "User ID Attribute Name" maps to the user ID lookup on the OAuth client side.

Certificate-Related Exceptions Communicating with External Social IdP

If you have any certificate-related exceptions communicating with an external social IdP, make sure you have performed the following:

- On a CA CloudMinder deployment, verify that the default CA certificate of the OAuth authorization server was imported.
- Import any other existing CA signed certificate that is required using the smkey tool provided.

Issues Passing Application Attributes After Successful SSO

If you have issues passing application attributes after a successful SSO, refer to the Application Attribute Definitions settings in Partnership.

SSO Fails with Social IdP

If the SSO fails with social IdP, review the log files:

- For SPS, check **FWSTRACE.log**: This file contains the log trace at the OAuth Consumer Service
- For SMPS

Chapter 15: Assertion Processing Customization (Relying Party)

The message consumer plug-in is a Java program that implements the Message Consumer Extension API. The plug-in lets you implement your own business logic for processing assertions, such as rejecting an assertion and returning a status code. This additional processing works together with the standard processing of an assertion.

During authentication, the system first tries to process the assertion by mapping a user to its local user store. If CA SiteMinder® Federation cannot find the user, it calls the postDisambiguateUser method of the message consumer plug-in.

If the plug-in successfully finds the user, the process continues to the second phase of authentication. If the plug-in cannot map the user to a local user store, the plug-in returns a UserNotFound error. The plug-in can optionally use the redirect URL feature. Without the consumer plug-in, the redirect URLs are based on the error that the SAML authentication scheme generates.

During the second phase of authentication, the system calls the postAuthenticateUser method of the message consumer plug-in, if the plug-in is configured. If the method succeeds, CA SiteMinder® Federation redirects the user to the requested resource. If the method fails, you can configure the plug-in to send the user to a failure page. The failure page can be one of the redirect URLs that you can specify with the authentication scheme configuration.

Reference information (method signatures, parameters, return values, data types), and the constructor for UserContext class, are in the *Java SDK Programming Reference*. Refer to the MessageConsumerPlugin interface.

To configure the plugin:

1. Install the CA SiteMinder® Federation SDK.
2. Implement the MessageconsumerPlugin.java interface, which is part of the SDK.
3. Deploy your message consumer plug-in implementation class.
4. Enable the message consumer plug-in in the Administrative UI.

Implement the MessageConsumerPlugin Interface

Create a custom message consumer plug-in by implementing the MessageConsumerPlugin.java interface. The minimum requirements for the implementation class are listed in the following procedure.

Follow these steps:

1. Provide a public default constructor method that contains no parameters.
2. Provide code so that the implementation is stateless. Many threads must be able to use a single plug-in class.
3. Implement methods in the interface as your requirements demand.

   The MessageConsumerPlugin includes the following four methods:

   init()

   Performs initialization procedures that the plug-in requires. CA SiteMinder calls this method once for each plug-in instance, when the plug-in is loaded.

   release()

   Performs any rundown procedures that the plug-in requires. CA SiteMinder calls this method once for each plug-in instance, when CA SiteMinder is shutting down.

   postDisambiguateUser()

   Provides processing to disambiguate a user when the authentication scheme is unable to do so. Alternatively, this method can add data for new federation users to a user store. This method receives the decrypted assertion. The decrypted assertion is added to the properties map passed to plug-in under the key "_DecryptedAssertion".

   postAuthenticateUser()

   Provides additional code to determine the outcome of assertion processing, regardless of whether the Policy Server processing is a success or failure.

The product provides the following samples of the Message Consumer plug-in class:

- MessageConsumerPluginSample.java
- MessageConsumerSAML20.java

The default location for the samples is:

Windows

C:\Program Files\FederationManager\sdk\java\sample

The package name is com\ca\federation\sdk\plugin\sample.

UNIX

/FederationManager/sdk/java/sample

The package name is com/ca/federation/sdk/plugin/sample.

Deploy a Message Consumer Plug-in

After you have coded your implementation class for the MessageConsumerPlugin interface, compile it and verify that CA SiteMinder® Federation can find your executable file.

Follow these steps:

1. Compile the MessageConsumerPlugin Java file. The file requires the following dependent libraries, which are installed with the product:

   federation_install_dir\siteminder\bin\jars\SmJavaApi.jar

   federation_install_dir is the directory where you installed CA SiteMinder® Federation

2. When a plug-in class is available, in a folder or a jar file, modify the -Djava.class.path value in the JVMOptions.txt file. This step enables the plug-in class to load with the modified classpath.

   Locate the JVMOptions.txt file in the directory *federation_mgr_installation_home*\siteminder\config.

   Note: Do not modify the classpath for the existing xerces.jar, xalan.jar, or SmJavaApi.jar.

3. Restart the system to pick up the latest version of MessageConsumerPlugin. This step is necessary each time the plug-in Java file is recompiled.

4. Enable the plug-in.

Enable the Message Consumer Plug-in in the UI

After writing a message consumer plug-in and compiling it, enable the plug-in by configuring settings in the Administrative UI. The UI settings tell CA SiteMinder® Federation where to find the plug-in.

Do not configure the plug-in settings until you deploy the plug-in (see page 179).

To enable the message consumer plug-in

1. Log on to the Administrative UI.

   Select the Consumer-to-Producer or SP-to-IdP partnership that you want to modify.
2. Navigate to the User Identification step in the partnership wizard.
3. In the Message Consumer Plug-in section, complete the following fields:

   Plug-in Class

   Specify the Java class name for the plug-in, For example, a sample class included with the SDK is:

   com.ca.messageconsumerplugin.MessageConsumerPluginSample

   Plug-in Parameters

   Specify a string of parameters that are passed to the plug-in specified in the Full Java Class Name field.
4. Restart the federation services according to your operating environment.

   - **Windows**

     Use the stop and start shortcuts as follows. If you logged in as a network user and not a local administrator, right-click the shortcut and select Run as administrator.

     a. Start, All Programs, CA, Federation Standalone, Stop services

     b. Start, All Programs, CA, Federation Standalone, Start services

   - **UNIX**

     a. Open a command window.

     b. Run the following scripts:

     federation_install_dir/fedmanager.sh stop

     federation_install_dir/fedmanager.sh start

     Note: Do not stop and start the services as the root user.

Chapter 16: Delegated Authentication

Delegated Authentication Overview

One of the configuration decisions for single sign-on is determining how users are authenticated.

CA SiteMinder offers two authentication choices:

- Local authentication

  CA SiteMinder authenticates the user at the local site. You configure an authentication URL in the Administrative UI where the user is redirected to authentication and to establish a session.

- Delegated authentication

  CA SiteMinder uses a third-party web access management (WAM) application that CA SiteMinder does not protect. The third-party application authenticates any user who requests a protected federated resource then forwards the federated user identity to CA SiteMinder. After CA SiteMinder receives the user identity information, it locates the user in its own user directory and starts the federation process with the relying party.

A delegated authentication request takes place at the asserting party and it can be initiated at the third-party WAM system or at CA SiteMinder. An authentication request can initiate at the relying party; however this scenario is not considered delegated authentication.

Authentication can be initiated as follows:

Authentication Initiated by CA SiteMinder at the Asserting Party

CA SiteMinder can initiate an authentication request at an asserting party. If the request is made to CA SiteMinder, it is recognized as a delegated authentication request. CA SiteMinder then redirects the user to the third-party WAM system.

Authentication Initiated by Direct Login to the WAM System at the Asserting Party

When a user logs in to a WAM system at the asserting party, an authentication request is initiated. After the WAM system successfully authenticates the user, the identity information is then forwarded to CA SiteMinder.

Authentication Initiated at the Relying Party

The relying party can initiate an authentication request, but this scenario is not considered delegated authentication. Delegated authentication occurs only at the asserting party.

A request for a federated resource is made directly to the relying party, who then sends an AuthnRequest to CA SiteMinder at the asserting party. CA SiteMinder recognizes it as a delegated authentication request and redirects the user to the third-party WAM system at the asserting party. The user logs in to the WAM system, which initiates an authentication request. After the WAM system successfully authenticates the user, the identity information is then forwarded to CA SiteMinder.

After the third-party WAM system receives the authentication request, it passes the user identity to CA SiteMinder. The method the WAM system uses to pass the user identity depends on whether the delegated authentication method is cookie-based or a query string-based.

How the Third Party WAM Passes the User Identity

The third-party WAM system can use one of two methods to pass a federated user identity to CA SiteMinder:

- Using an open format cookie.

  You can encrypt the open format cookie to help ensure the security of the data.

- Using a query string that is appended to a redirect URL that sends the browser to CA SiteMinder.

  The query string is sent in clear text.

  Important! Do not use the query string method in a production environment. The query string redirection method is only for a testing environment as a proof of concept.

The method a third-party WAM system chooses depends on the configuration it wants to establish for passing a user identity to CA SiteMinder.

The methods of passing the user identity are detailed in the following sections.

Cookie Method for Passing User Identity

CA SiteMinder can use an open-format cookie to pass a user identity. The cookie contains a user login ID as one of its values.

Authentication can begin at the WAM system or at CA SiteMinder. If authentication begins at CA SiteMinder, it redirects the user to the WAM system. The authentication process is the same as if it began at the WAM system.

The delegated authentication process is as follows:

1. An authentication request comes into to the third-party WAM system.
2. The user is authenticated.
3. The third-party WAM system obtains a cookie in one of two ways:
   - The WAM system uses the CA SiteMinder® Federation SDK to create an open-format cookie. The SDK creates the cookie and sends it back in a request to the WAM system.

     Note: To create an open-format cookie that is FIPS-encrypted, use a CA SiteMinder® Federation SDK.

     The third-party WAM application uses the same language as the SDK that it is using to create a cookie. If you are using the CA SiteMinder® Federation Java SDK, the third-party WAM application must be in Java. If you are using the .NET SDK, the third-party WAM application must support .NET.
   - The WAM system uses a manually created open-format cookie.

     You can create an open-format cookie without using a CA SiteMinder® Federation SDK. To create the cookie manually, use any programming language that supports UTF-8 encoding. You can use any of the following PBE encryption algorithms that CA SiteMinder supports for password-based encryption:
     - PBE/SHA1/AES/CBC/PKCS12PBE-1000-128
     - PBE/SHA1/AES/CBC/PKCS12PBE-1000-192
     - PBE/SHA1/AES/CBC/PKCS12PBE-1000-256
     - PBE/SHA256/AES/CBC/PKCS12PBE-1000-128
     - PBE/SHA256/AES/CBC/PKCS12PBE-1000-192
     - PBE/SHA256/AES/CBC/PKCS12PBE-1000-256
     - PBE/SHA1/3DES_EDE/CBC/PKCS12PBE-1000-3
     - PBE/SHA256/3DES_EDE/CBC/PKCS12PBE-1000-3

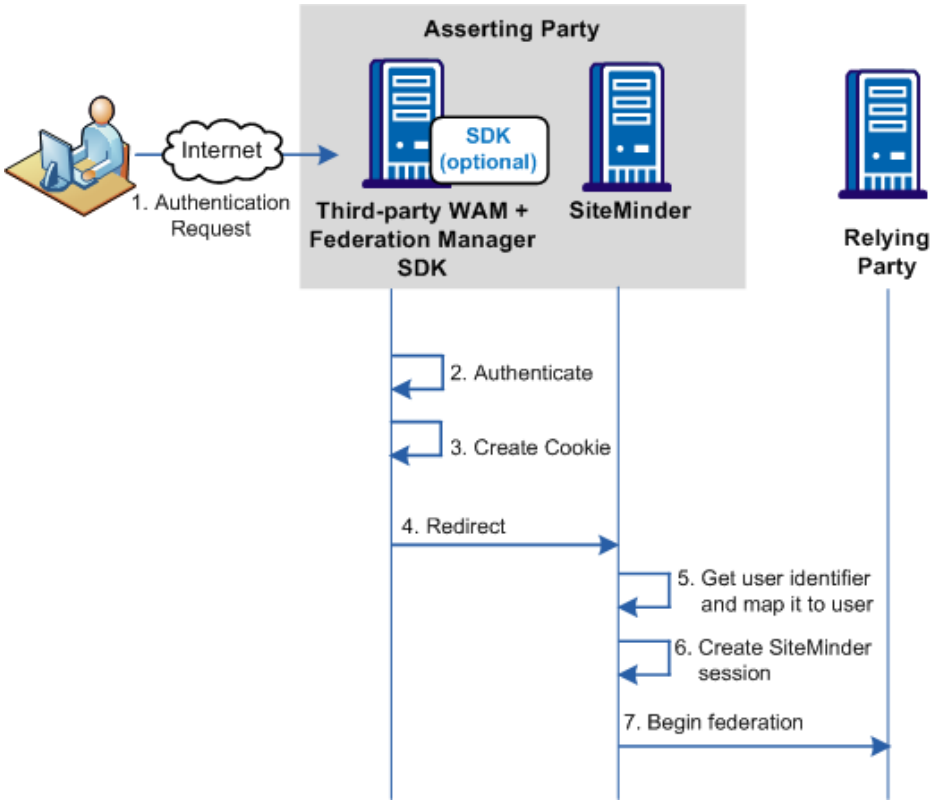
     Verify that the open-format cookie gets set in the browser.

     To write a complete cookie, review the details about the contents of the open-format cookie.

   Note: The WAM system and CA SiteMinder must be in the same cookie domain.

4. The WAM system redirects the browser to CA SiteMinder.
5. CA SiteMinder extracts the login ID from the cookie then locates the user in its user directory.
6. CA SiteMinder creates a CA SiteMinder session.
7. After the session is created, federated communication with the relying party proceeds.

The following picture shows the cookie method when authentication is initiated at the third-party WAM. CA SiteMinder is not protecting the WAM application.



Important! To use an SDK-created open-format cookie, the third party must install a CA SiteMinder® Federation SDK. The SDK is a separately installed component from CA SiteMinder. The installation kit contains the documentation that describes how to use the SDK for delegated authentication.

Query String Method for Passing User Identity

A third-party WAM system can pass a user identity to CA SiteMinder by appending a query string on the redirect URL. For this method to work, the third-party WAM system has to configure a URL that redirects federated users to CA SiteMinder after they are authenticated.

Important! Do not use the query string method in a production environment. The query string redirection method is only for a testing environment as a proof of concept.

If authentication is initiated at the WAM system, the process for delegated authentication using a query string is as follows:

Note: Authentication can also be initiated at CA SiteMinder or at the relying party.

1. The third-party WAM system receives an authentication request.
2. The user is authenticated.
3. The third-party WAM system constructs a redirect URL and adds the login ID and hashed login ID values to the query string in the format LoginID=*LoginID*&LoginIDHash=*hashed_LoginID.*

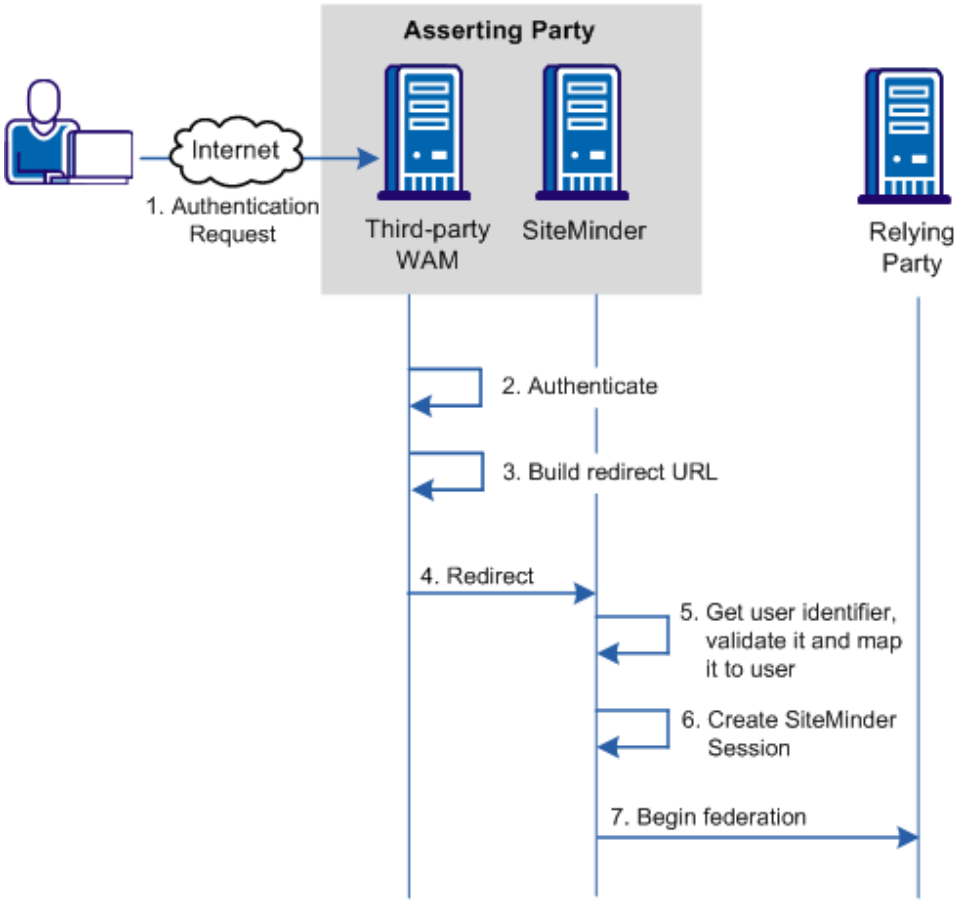
   Important! The LoginID and LoginIDHash parameters are case-sensitive. Be sure to include them in the redirect URL as shown in the example.

   The hashing mechanism allows CA SiteMinder to verify that the user ID has been received unchanged.

   Example of a Redirect URL

   http://idp1.example.com:9090/affwebservices/public/saml2sso?SPID=FmSP&ProtocolBinding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST&LoginID=jdoe&LoginIDHash=454d3bd5cb839168eeffcf060ae0b9c28ed6eec0
4. The WAM system redirects the browser to CA SiteMinder.
5. CA SiteMinder extracts the login ID and hashed login ID from the URL, validates the identifier using the hashed value, and locates the user in its user directory.
6. CA SiteMinder creates a user session.
7. After the session is created, federated communication with the relying party proceeds.

The following picture shows the query string method when authentication is initiated at the asserting party.



Delegated Authentication Configuration

Delegated authentication is configured at the asserting party, where an assertion is generated based on an authenticated user identity.

To configure delegated authentication

1. Determine which method (cookie or query string) the third-party WAM uses to pass the user identity.

   Note: The query string does not produce a FIPS-compliant partnership.

2. Go to the appropriate step in the partnership wizard to set up delegated authentication.

Important! To use the SDK-created open-format cookie, the third party must install a CA SiteMinder® Federation SDK. The SDK is a separately installed component. The installation kit contains the documentation that describes how to use the SDK for delegated authentication.

Cookie Delegated Authentication Sample Setup

The following sample configuration is from the perspective of a SAML 2.0 IdP > SP partnership. The delegated authentication settings are on the SSO and SLO step of the partnership wizard.

This sample configuration reflects a SAML 2.0 configuration. The Identity Provider is http://idp1.xyz.com and the third-party WAM system is http://wamservice.xyz.com.

To configure cookie delegated authentication

1. Create a partnership or edit an existing one.

   Note: To edit a partnership, deactivate it first.

2. Navigate to the SSO and SLO step in the Partnership wizard.

3. In the Authentication section, set the fields as follows:

   Authentication Mode

   Delegated

   Delegated Authentication Type

   Open format cookie

   For use with a web access management application. You can use a CA SiteMinder® Federation SDK to create a Java or .NET application. Alternatively, you can use an application written in another language, provided you build the open-format cookie manually.

   If you require FIPS 140-2 encryption, create the open-format cookie using the CA SiteMinder® Federation Java or .NET SDK.

   Delegated Authentication URL

   http://wamservice.xyz.com

   The URL of the third-party WAM system that authenticates users and uses a CA SiteMinder® Federation SDK to create the cookie.

   Authentication Class

   Enter the authentication method that is used at the third party. For example:

   urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos

4. Communicate all the open-format cookie settings to the third-party WAM system.

   CA SiteMinder uses these values in the creation of the cookie.

5. Continue with partnership configuration.

Query String Delegated Authentication Sample Setup

The following sample configuration is from the perspective of a SAML 2.0 IdP > SP partnership. The delegated authentication settings are on the SSO and SLO step of the partnership wizard.

Note: The query string method does not produce a FIPS-compliant partnership.

This sample configuration reflects a SAML 2.0 configuration. The Identity Provider is http://idp1.xyz.com and the third-party WAM system is http://wamservice.xyz.com.

Important! Do not use the query string method in a production environment. The query string redirection method is only for a testing environment as a proof of concept.

To configure query string delegated authentication

1. Create a partnership or edit an existing one.

   Note: To edit a partnership, deactivate it first.

2. Navigate to the appropriate step in the partnership wizard.

3. In the Authentication section, set the fields as follows:

   Authentication Mode

   Delegated

   Delegated Authentication Type

   Query String

   Delegated Authentication URL

   http://wamservice.xyz.com

   The URL of the third-party WAM system that authenticates users and constructs the redirect URL back to CA SiteMinder with the query parameters.

   Hash Secret

   FederatedAuth1

   The third-party WAM system uses this secret to hash the login ID.

   Confirm Hash Secret

   FederatedAuth1

   Authentication Class

   Enter the authentication method that is used at the third party. For example:

   urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos

4. Continue with partnership configuration.

Third-party WAM Configuration for Cookie Delegated Authentication

For delegated authentication to succeed, the third-party WAM must adjust its federated application, as follows:

- To communicate the authenticated user login ID through a cookie, the third-party WAM system must generate a cookie.
  - For Java applications, the WAM can use a CA SiteMinder® Federation Java SDK to create a legacy cookie or an open-format cookie.
  - For .NET applications, the WAM can use a CA SiteMinder® Federation .NET SDK to create an open-format cookie.
  - For languages other than Java and .NET, the WAM can create an open-format cookie manually.

  For details on implementing the necessary class and methods, see the *CA SiteMinder® Federation Java SDK Guide* or the *CA SiteMinder® Federation .NET SDK Guide.* Each guide is installed with the SDK. If you create an open-format cookie manually, review the details about the required contents of the cookie.
- The third party must know the values of the following Administrative UI settings that are configured at the CA SiteMinder asserting party:
  - Encryption Password
  - Open-format Cookie Name
  - Open-format Cookie Encryption Transformation

  CA SiteMinder uses these values when creating the cookie. These settings are on the Single Sign-On (SAML 1.x) and SSO and SLO (SAML 2.0) steps of the partnership wizard.
- The third-party WAM system must create a redirect URL that sends the user back to CA SiteMinder. This URL has to send the user back to the CA SiteMinder single sign-on service. The CA SiteMinder Administrator has to tell the third party about this URL in an out-of-band communication.

  Important! After the third-party WAM system receives an authentication request from CA SiteMinder, it must capture and resend any existing query string. The incoming request can have CA SiteMinder request information within the query string and the request must pass unchanged.

  Note: To pass the cookie, the third-party WAM system must be in the same cookie domain as CA SiteMinder at the asserting party.

Third-party WAM Configuration for Query String Delegated Authentication

A third-party WAM system and CA SiteMinder at the asserting party communicate the login ID in a query string. The WAM system must add the following two attributes to the query string in the redirect URL:

LoginID

Specifies the value that identifies the user to the third-party WAM system.

Important! The LoginID parameter is case-sensitive.

LoginIDHash

A hash of the LoginID.

To generate the LoginIDHash value, the LoginID is prepended to a Hash Secret and the entire value is then run through a SHA-1 hashing algorithm. The Hash Secret is specified in the CA SiteMinder configuration at the asserting party.

When CA SiteMinder retrieves the credentials from the query string, it also combines these values and hashes them. If the hashes are equal, CA SiteMinder considers the login ID to be valid and continues with the federation request.

Important! The LoginIDHash parameter is case-sensitive.

The third-party WAM system must configure its federated application to construct a redirect URL that sends the user back to the CA SiteMinder Single Sign-on service. Therefore, the CA SiteMinder Administrator has to communicate the Single Sign-on service to the third party in an out-of-band communication.

Important! After the third-party WAM system receives an authentication request from CA SiteMinder, it captures and resends any existing query string. If the incoming request has CA SiteMinder request information within the query string, the WAM system must pass it along unchanged.

The syntax of the query string is as follows:

?*existing_query_string*&LoginID=*LoginID*&LoginIDHash=*hashed_LoginID*

Example

```
https://johndoe3227.b.com/affwebservices/public/saml2sso?SPID=sp1&
LoginID=user1&LoginIDHash=de164152ed6e8e9a7f760e47d135ecf0c98a
3e4e&ProtocolBinding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact
```

Chapter 17: URLs to Initiate Single Sign-on

Links to Servlets which Initiate Single Sign-on

When designing a site for federated content, that site includes a page with specific links to trigger single sign-on. These links are URLs to servlets for the Single Sign-on service or the AuthnRequest Service.

To initiate single sign-on, the user can begin at the asserting or relying party. Configure the appropriate links at each site to initiate single sign-on operation.

Producer-initiated SSO (SAML 1.1)

At the producer, create pages that contain links that direct the user to the consumer site. Each link represents an intersite transfer URL. The user has to visit the intersite transfer URL. The URL makes a request to the producer-side web Agent before the user is redirected to the consumer site.

For SAML Artifact and POST profile, the syntax for the intersite transfer URL is:

`http://`*`producer_host:port`*`/affwebservices/public/intersitetransfer?`
`CONSUMERID=`*`consumer_entity_ID`*`&TARGET=`*`http://consumer_site/target_url`*

The variables and query parameters in the previous intersite transfer URL are as follows:

producer_host:port

Specifies the server and port number where the user is authenticated.

CONSUMERID

(Required) Identifies the consumer. On the producer side, the producer-to-consumer partnership has a name, and the remote consumer entity has an ID. The CONSUMERID is the entity ID of the remote consumer. The entity ID is case-sensitive. Enter it exactly as it appears in the Administrative UI.

You can use the parameter NAME in place of CONSUMERID, but not both.

If you use NAME, specify the name of the producer-to-consumer partnership as defined at the producer.

consumer_entity_ID

Identifies the consumer site the user wants to visit from the producer site. The entity ID is case-sensitive. Enter it exactly as it appears in the Administrative UI.

TARGET

(Optional) Identifies the requested target resource at the consumer.

The TARGET parameter is optional. You are required to define the target; however, you can define it in the consumer-side partnership instead of the intersite transfer URL. The target is defined in the Application Integration step of the Partnership wizard. Be sure to define the target in the URL or in the partnership.

consumer_site

Specifies the server at the consumer site.

target_url

Indicates the target application at the consumer site.

Note: Query parameters for the SAML Artifact binding must use HTTP-encoding.

Example of an intersite transfer URL for the Artifact and POST profile:

```
http://www.smartway.com/affwebservices/public/intersitetransfer?
CONSUMERID=ahealthco&TARGET=http://www.ahealthco.com:85/
smartway/index.jsp
```

IdP-initiated SSO (SAML 2.0 Artifact or POST)

If a user visits a CA SiteMinder Identity Provider before going to the Service Provider, an unsolicited response at the Identity Provider must be initiated. To initiate an unsolicited response, create a hard-coded link that generates an HTTP Get request that CA SiteMinder accepts. This HTTP Get request must contain a query parameter that provides the Service Provider ID. The Identity Provider must generate the SAML assertion response. A user clicks this link to initiate the unsolicited response.

Note: This information applies to Artifact or POST bindings.

To specify the use of artifact or POST profile in the unsolicited response, the syntax for the unsolicited response link is:

http://*idp_server:port*/affwebservices/public/saml2sso?SPID=*SP_ID*&
ProtocolBinding=*URI_for_binding*&RelayState=*target_URL*

idp_server:port

Identifies the web server and port hosting CA SiteMinder.

SP_ID

Specifies the Entity ID of the Service Provider defined in the partnership. The entity ID is case-sensitive. Enter it exactly as it appears in the Administrative UI.

URI_for_binding

Identifies the URI of the POST or Artifact binding for the ProtocolBinding element. The SAML 2.0 specification defines this URI.

- The URI for the artifact binding, as specified by the SAML 2.0 specification is:

  urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact

- The URI for the POST binding, as specified by the SAML 2.0 specification is:

  urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST

  You do not need to set this parameter for HTTP-POST single sign-on.

Note: A binding must also be enabled for the partnership for the request to work.

target_URL

Specifies the URL of the federation resource target at the Service Provider.

Note the following:

- If you do not include the ProtocolBinding query in the link, use the one binding configured in the Service Provider properties
- When Artifact and POST are enabled in the Service Provider properties, POST is the default. Therefore, if you only want to use Artifact binding, include the ProtocolBinding query parameter in the link.

Important! If you configure indexed endpoint support for Assertion Consumer Services, the value of the ProtocolBinding query parameter overrides the binding for the Assertion Consumer Service.

Unsolicited Response Query Parameters Used by the IdP

An unsolicited response that initiates single sign-on from the IdP can include the following query parameters:

SPID

(Required) Specifies the ID of the Service Provider where the Identity Provider sends the unsolicited response. The entity ID is case-sensitive. Enter it exactly as it appears in the Administrative UI.

ProtocolBinding

Specifies the ProtocolBinding element in the unsolicited response. This element specifies the protocol for sending the assertion response to the Service Provider. If the Service Provider is not configured to support the specified protocol binding, the request fails.

RelayState

Indicates the URL of the target resource at the Service Provider. By including this query parameter, it tells the IdP to redirect the user the appropriate resource at the Service Provider. This query parameter can be used in place of specifying a target URL when configuring single sign-on.

Required Use of the ProtocolBinding Query Parameter

The ProtocolBinding query parameter is required *only* if the artifact and POST binding are enabled for the Service Provider properties. In addiiton, the user wants to only use artifact binding.

- The URI for the artifact binding is:

  urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact

- The URI for the POST binding is:

  urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST

  You do not need to set this parameter for HTTP-POST single sign-on.

Note: HTTP coding the query parameters is not necessary.

Optional Use of the ProtocolBinding Query Parameter

When you *do not* use the ProtocolBinding query parameter, the following information applies:

- If only one binding is enabled for the Service Provider and the ProtocolBinding is not specified in the unsolicited response, the enabled binding is used.
- If both bindings are enabled for the Service Provider and the ProtocolBinding is not specified in the unsolicited response, the POST binding is the default.

Example: Unsolicited Response without ProtocolBinding

The link redirects the user to the Single Sign-on service. Included in this link is the Service Provider identity, which the SPID query parameter specifies. The ProtocolBinding query parameter is not present. After the user clicks this hard-coded link, they are redirected to the Single Sign-on service.

http://fedsrv.fedsite.com:82/affwebservices/public/saml2sso?
SPID=http%3A%2F%2Ffedsrv.acme.com%2Fsmidp2for90

Example: Unsolicited Response with ProtocolBinding

The link redirects the user to the Single Sign-on service. Included in this link is the Service Provider identity, which the SPID query parameter specifies and the artifact binding is being used. After the user clicks this hard-coded link, they are redirected to local Single Sign-on service.

http://idp-ca:82/affwebservices/public/saml2sso?SPID=
http%3A%2F%2Ffedsrv.acme.com%2Fsmidp2for90&
ProtocolBinding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact

ForceAuthn and IsPassive Processing at the IdP

If Service Provider initiates single sign-on, the Service Provider can include a ForceAuthn or IsPassive query parameter in an AuthnRequest message.

When a Service Provider includes ForceAuthn or IsPassive in the AuthnRequest, a CA SiteMinder Identity Provider handles these query parameters as follows:

ForceAuthn Handling

When a Service Provider includes ForceAuthn=True in the AuthnRequest message, a CA SiteMinder Identity Provider challenges the user for their credentials. The challenge even when a CA SiteMinder session exists.

IsPassive Handling

A CA SiteMinder IdP does not support passive authentication. When a Service Provider includes IsPassive in the AuthnRequest and the Identity Provider cannot honor it, the IdP sends back one of these SAML responses:

- If IsPassive=True in the AuthnRequest message and there is no session, the Identity Provider returns an error message. CA SiteMinder requires a session.
- If IsPassive=True in the AuthnRequest message and there is a session, the Identity Provider returns the assertion.
- If IsPassive and ForceAuthn are in the AuthnRequest message and both are set to True, the CA SiteMinder Identity Provider returns an error. IsPassive and ForceAuthn are mutually exclusive.

SP-initiated SSO (SAML 2.0)

SP-initiated SSO requires that you have an HTML page at the Service Provider containing hard-coded links to the AuthnRequest service at the Service Provider. The links redirect the user to the Identity Provider to be authenticated and determining what is included in the AuthnRequest itself.

This information applies to Artifact or POST bindings.

The hard-coded link that the user selects must contain specific query parameters, which are used in an HTTP GET request to the AuthnRequest service.

Note: The page with these hard-coded links has to reside in an unprotected realm.

To specify the use of artifact or profile binding for the transaction, the syntax for the link is:

http://*sp_server:port*/affwebservices/public/saml2authnrequest?
ProviderID=*IdP_ID*&ProtocolBinding=*URI_of_binding*&
RelayState=*target_URL*

sp_server:port

Specifies the server and port number at the Service Provider that is hosting CA SiteMinder® Federation.

IdP_ID

Specifies the identity that is assigned to the Identity Provider. The entity ID is case-sensitive. Enter it exactly as it appears in the Administrative UI.

URI_of_binding

Identifies the URI of the POST or Artifact binding for the ProtocolBinding element. The SAML 2.0 specification defines this URI.

- The URI for the artifact binding is:

  urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact

- The URI for the POST binding is:

  urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST

  You do not need to set this parameter for HTTP-POST single sign-on.

Also, enable a binding for the partnership for the request to work.

target_URL

Specifies the URL of the federation target at the Service Provider.

Note the following information:

- If you do not include the ProtocolBinding query parameter in the AuthnRequest link, the default binding is the one defined for the partnership. If you have both bindings defined in the partnership, then no binding is passed in the AuthnRequest. As a result, the default binding at the Identity Provider is used.
- If the artifact and POST bindings are enabled for the partnership but you only want to use artifact binding, include the ProtocolBinding query parameter in the link.

AuthnRequest Query Parameters Used by an SP

The query parameters a CA SiteMinder SP can use in the links to the AuthnRequest Service are as follows:

ProviderID (required)

Entity ID of the Identity Provider where the AuthnRequest Service sends the AuthnRequest message. The entity ID is case-sensitive. Enter it exactly as it appears in the Administrative UI.

ProtocolBinding

Specifies the ProtocolBinding element in the AuthnRequest message. This element specifies the protocol for returning the SAML response from the Identity Provider. If the specified Identity Provider is not configured to support the specified protocol binding, the request fails.

If you use this parameter in the AuthnRequest, you cannot include the AssertionConsumerServiceIndex parameter also. They are mutually exclusive.

ForceAuthn

Instructs the Identity Provider that it must authenticate a user directly instead of relying on an existing security context. Use this query parameter when the Identity Provider is using CA SiteMinder® Federation, not if it is using third-party federation software.

- If the SP sets ForceAuthn=True in the AuthnRequest message, and a session exists for a particular user, the Identity Provider challenges the user. If the user successfully authenticates, the IdP sends the identity information from the existing session in the assertion. The Identity Provider discards the session that it generates for the reauthentication.
- If the SP sets ForceAuthn=True in the AuthnRequest message and there is no session, the IdP challenges the user. If the user successfully authenticates, a session is established.

Example

http://sp1.demo.com:81/affwebservices/public/saml2authnrequest?ProviderID=idp1.example.com&ForceAuthn=yes

IsPassive

Instructs the Identity Provider to log in the user without challenging the user for credentials or interacting with the user in any way. A CA SiteMinder Identity Provider does not honor this query parameter unless the user has a session. If the user does not have a session, the Identity Provider returns an error.

AssertionConsumerServiceIndex

Specifies the index of the endpoint acting as the Assertion Consumer Service. The index tells the Identity Provider where to send the assertion response.

If you use this parameter in the AuthnRequest, do not include the ProtocolBinding parameter also. This parameter and the ProtocolBinding parameter are mutually exclusive. The Assertion Consumer Service has its own protocol binding, which could conflict with the ProtocolBinding parameter.

RelayState

Indicates the URL of the target resource at the Service Provider. By including this query parameter, it tells the Service Provider where to send the user. Otherwise, the default target for the partnership is used.

Required Use of the ProtocolBinding Query Parameter

The ProtocolBinding parameter is required if the artifact and POST bindings are enabled for the partnership, and the user wants to use only the artifact binding.

- The URI for the artifact binding is:

  urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact

- The URI for the POST binding is:

  urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST

  You do not need to set this parameter for HTTP-POST single sign-on.

Optional Use of ProtocolBinding

If you *do not* use the ProtocolBinding query parameter the following conditions apply:

- If only one binding is enabled for the partnership and the ProtocolBinding query parameter is not specified, the enabled binding for the partnership is used.
- If both bindings are enabled and the ProtocolBinding query parameter is not specified, POST binding is used as the default.

Note: You do not need to HTTP-encode the query parameters.

Example: AuthnRequest Link without the ProtocolBinding Query Parameter

This sample link goes to the AuthnRequest service. The link specifies the Identity Provider in the ProviderID query parameter.

http://ca.sp.com:90/affwebservices/public/saml2authnrequest?
ProviderID=http%3A%2F%2Ffedsrv.acme.com%2Fsmidp2for90

After a user clicks the link at the Service Provider, CA SiteMinder passes a request for an AuthnRequest message.

Example: AuthnRequest Link with the ProtocolBinding Query Parameter

http://ca.sp.com:90/affwebservices/public/saml2authnrequest?
ProviderID=http%3A%2F%2Ffedsrv.acme.com%2Fsmidp2for90&
ProtocolBinding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact

After a user clicks the link at the Service Provider, CA SiteMinder passes a request for an AuthnRequest message.

IP-initiated Single Sign-on (WSFED)

A user can visit the Identity Provider (IP) before going to the Resource Partner (RP). If the user visits the Identity Provider first, a link must generate an HTTP Get request. The hard-coded link points to the passive requester service at the IP. The request contains the RP Provider ID and optionally other parameters.

The syntax for the link is:

https://i*p_server:port*/affwebservices/public/wsfedsso?wa=wsignin1.0&wtrealm=*rp_id*

ip_server:port

Specifies the server and port number of the system at the Identity Partner. The system is hosting the Web Agent Option Pack or the SPS federation gateway, depending on which component is installed in your federation network.

rp_id

The ID of the RP. The entity ID is case-sensitive. Enter it exactly as it appears in the Administrative UI.

RP-initiated Single Sign-on (WSFED)

When a user starts at the RP to initiate single sign-on, typically the user selects from a list of IPs. The site selection page is in an unprotected realm.

The link on the site selection page points to the passive requester service at an IP. After the link is selected, the RP redirects the user to the IP to get the assertion.

Chapter 18: Logging Out of User Sessions

Single Logout Overview (SAML 2.0)

Single logout (SLO) results in the simultaneous termination of all user sessions for the browser that initiated the logout. Closing all user sessions prevents unauthorized users from gaining access to resources at the SPs.

Single logout does not necessarily end all sessions for a user. For example, a user with two browsers open can have two independent sessions. Only the session for the browser that initiates the logout is terminated at all federated sites for that session. The session in the other browser is still active.

The single logout binding determines what is sent with a single logout message and how each received message is handled.

Important! To configure single logout, enable the session store using the Policy Server Management Console. For instructions about using the Management Console, see the *Policy Server Administration Guide* for instructions.

Two bindings are available for single logout operation:

HTTP-Redirect

HTTP-Redirect binding relies on a browser to conduct each logout transaction. The single logout message is always a GET request. The browser is involved in every request and response. The involvement of the browser means that HTTP-redirect binding provides browser session data, which the SOAP binding does not.

A disadvantage of HTTP-Redirect binding is that the data in the message is limited to what you can send on the query string. Also, HTTP-Redirect binding is an asynchronous process so timeouts are unlikely. However, if a redirect fails, that failure stops the entire single logout chain.

SOAP

SOAP binding uses POST requests to conduct single logout transactions. POST requests let you send more data than the HTTP-Redirect binding. SOAP also enables you to do more in the way of encryption and other features.

SOAP is a synchronous process. The IdP has more control and can prevent a problem at a single SP from interfering with the whole process. SOAP communication takes place over a back channel. One logout failure does not have to stop the IdP from attempting to log out from the rest of the SPs.

SOAP relies on a back channel connection, so after the initial single logout call and response a browser is not involved. The SOAP binding does not clean up cookies at the remote entity as part of the logout process. Cookies are cleaned up only at the local entity. If deleting cookies is required, use HTTP-Redirect binding.

Managing Single Logout Across a Network Using HTTP-Redirect and SOAP

Your network can have some sites that support the HTTP-Redirect binding and others that support the SOAP binding. The IdP has to manage multiple bindings, but the SP sends or receives only one logout request.

The following sections provide configuration guidelines to handle a mixed-binding environment.

SLO Configuration when CA SiteMinder is at the IdP

When CA SiteMinder is at the IdP, configure the partnership to include an HTTP Redirect-based SLO Service URL and a SOAP-based SLO Service URL.

CA SiteMinder at the IdP inspects the configuration for each SP in a session and handles all SOAP-enabled logouts first. HTTP-Redirect logouts for SPs that do not support SOAP follow.

SLO Configuration when CA SiteMinder is at the SP

If CA SiteMinder is at the SP and the SP initiates single logout, we recommend the HTTP-Redirect binding to initiate the logout. Other SPs for the user session possibly do no support SOAP.

HTTP-Redirect relies on a browser session to handle all redirections. For this reason, it sends the necessary data that the IdP must have to logout SPs that only support HTTP-Redirect. If the SP starts the process with HTTP-Redirect, the IdP can use SOAP with all SPs that support it. Switch to HTTP-Redirect binding for the remaining SPs.

If you initiate single logout with the SOAP binding, the browser session data is not present.

To help ensure an SP-initiated logout uses HTTP-Redirect, embed an HTTP-Redirect link that points to the SP' local servlet in a page or application. For CA SiteMinder, that link is:

http://s*p_host:port*/affwebservices/public/saml2slo

This embedded link causes CA SiteMinder to generate a SAML <LogoutRequest> message that it sends to the SLO service at the IdP. When a user logs out, the logout at the SP is performed first and then the logout request is sent to the IdP. The IdP then completes the logout process with all the other SPs involved in the user session.

Understanding Skew Time for SLO Request Validity

Two values are relevant when calculating how long the logout request is valid. These values are the IssueInstant value and the NotOnOrAfter value. In the SLO response, the single logout request is valid until the NotOnOrAfter value. When a single logout request is generated, CA SiteMinder takes its system time. The resulting time becomes the IssueInstant set in the request message. To determine when the logout request expires, CA SiteMinder takes its current system time and adds the Skew Time plus the SLO Validity Duration. The resulting time becomes the NotOnOrAfter value.

Note: Times are relative to GMT.

For example, a log out request is generated at the asserting party at 1:00 GMT. The Skew Time is 30 seconds and the SLO Validity Duration is 60 seconds. Therefore, the request is valid between 1:00 GMT and 1:01:30 GMT. The IssueInstant value is 1:00 GMT and the single logout request message is no longer valid 90 seconds afterward.

Configure Single Logout

Configuring single logout requires that you enable the session store using the Policy Server Management Console. For instructions about using the Management Console, see the *Policy Server Administration Guide* for instructions. If the session store is not enabled, you cannot see the single logout settings in the Administrative UI.

When configuring single logout, note the following information:

- If a partner receives a SAML <LogoutRequest> message using HTTP-Redirect, the response back to the sending party must use the HTTP-Redirect binding.
- If a partner receives SAML <LogoutRequest> message using SOAP, the response back to the sending party must be over SOAP.
- If a partner receives an SLO request over a binding it does not support, single logout fails.
- If a single logout user session includes partners using the HTTP-Redirect and SOAP bindings, configure CA SiteMinder to support both bindings. When the IdP proceeds with the logout, it logs out all SPs using SOAP then logs out all SPs using HTTP-Redirect binding.
- If a CA SiteMinder SP initiates single logout, start by using the HTTP-Redirect binding, even if the SP supports SOAP.

  Review guidelines for managing single logout in a mixed environment (see page 204) where SOAP and HTTP-Redirect are supported.

Follow these steps:

Note: The SLO configuration settings are the same at the IdP and SP.

1. Begin at the SSO and SLO step of the partnership wizard.
2. In the SLO section, select one or both SLO bindings.

   The SLO binding enables single logout and indicates the binding in use at the local entity. The SLO binding also indicates which binding the local entity accepts when it receives a single logout request.

   If you select SOAP, you can encrypt the Name ID in the SOAP message. Encryption options are set in the Signature and Encryption step of the partnership wizard.

   If you select SOAP as the binding, the Incoming and Outgoing Configuration for the Back Channel becomes active. SLO requests and responses are sent across a back channel. Each local partner can secure the back channel by requiring the remote partner to authenticate.

   More information can be found about the back channel settings for SLO (see page 207).
3. Configure any of the additional SLO settings:
   - SLO Confirm URL
   - SLO Validity Duration (Seconds)
   - Relay State overrides SLO Confirm URL

   Click Help for the field descriptions.
4. Complete the table for the SLO Service URLs. You must have at least one entry. Values that are defined for the selected remote entity are already entered in the table.

   The SLO Service URL initiates single logout, which then triggers the Policy Server to generate a SAML <LogoutRequest> message. In addition, the SLO Service URL tells the Policy Server where to send the logout request message.

   Specify a SLO service URL for each supported SLO binding, as follows:
   - HTTP-Redirect enabled—select one URL with HTTP-Redirect as the binding.
   - SOAP enabled—select one URL with SOAP as the binding.
   - Redirect and SOAP enabled—select two URLs, one set to HTTP-Redirect and one set to SOAP.

   Note: The Response Location URL field is optional.

Single logout configuration is complete.

Back Channel Configuration for Single Logout

Single logout using the SOAP binding sends logout requests and responses across a back channel. You can require an entity to authenticate to access the back channel. The back channel can also be secured using SSL, though SSL is not required.

Securing the back channel using SSL involves:

- Enabling SSL.

  SSL is not required for Basic authentication but you can use Basic over SSL. SSL is required for Client Cert authentication.

- Configure an incoming and outgoing back channel for the single logout communication exchange. The local entity has to be able to send messages over the outgoing channel and receive messages over the incoming channel.

  Note: You can configure an incoming and outgoing back channel; however, a channel can have only one configuration. If two services use the same channel, these two services use the same back channel configuration. For example, if the incoming channel for a local asserting party supports HTTP-Artifact SSO and SLO over SOAP, these two services must use the same back channel configuration.

- Choosing the type of authentication for the remote entity to gain access across the protected back channel. The authentication method applies per channel (incoming or outgoing).

  The options for back channel authentication are:

  Basic

  Indicates that a Basic authentication scheme is protecting the back channel.

  Note: If SSL is enabled for the back channel connection, Basic authentication can still be selected.

  Client Cert

  Indicates that SSL with an X.509 client certificate protects the asserting party back channel.

  If you select Client Cert as the authentication method, all endpoint URLs have to use SSL communication. This means that the URLs must begin with **https://**. Endpoint URLs locate the various SAML services on a server, such as single sign-on, single logout, and the Assertion Consumer Service.

NoAuth

Indicates that the relying party is not required to supply credentials. The back channel is not secured. You can still enable SSL with this option. The back channel traffic is encrypted but no credentials are exchanged between parties.

Use the NoAuth option for only for testing purposes, not for production. The exception is when CA SiteMinder sits behind a proxy server implementing SSL-enabled failover. If client certificate authentication is used to protect the back channel, the proxy server handles the authentication. All IdP->SP partnerships can use NoAuth as the authentication type.

Important! The authentication method for the incoming back channel must match the outgoing back channel on the other side of the partnership. Agreeing on the choice of authentication method is handled in an out of band communication.

To secure the back channel for single logout

1. Begin at the Back Channel section in the SSO and SLO step of the partnership wizard.
2. Select SOAP in the SLO section. The Authentication Method field becomes active.
3. Select the type of authentication method for the incoming and outgoing back channel. Additional fields to configure are displayed for Basic and Client Cert methods.

   If you select No Auth as the authentication method, no additional steps are required.
4. Depending on the authentication method you select, several additional fields are displayed for you to configure.

After entering values for all the necessary fields, the back channel configuration is complete.

Sign-Out Overview (WS-Federation)

Sign-out is the simultaneous termination of all user sessions for the browser that initiated the sign-out. Closing all user sessions prevents unauthorized users from gaining access to resources at the Resource Partner.

Sign-out does not necessarily end all sessions for a user. For example, a user with two browsers open can have two independent sessions. Only the session for the browser that initiates the sign-out is terminated at all federated sites for that session. The session in the other browser is still active.

The Policy Server performs sign-out using a signoutconfirmurl.jsp. This page resides on the Identity Provider system. An Identity Provider partner initiates a sign-out request on behalf of a user. The JSP sends the sign-out request to each site where the user signed on during a given browser session. The user is then signed out.

A user can initiate a sign-out request only at an Identity Provider. The request is triggered by clicking a link that points to the appropriate servlet. The sign-out confirmation page must be an unprotected resource at the Identity Provider site.

Note: The Policy Server only supports the WS-Federation Passive Request profile for sign-out.

Enable WSFED Sign-Out

Requirements to configure sign-out:

- To enable sign-out at the Identity Provider, enable the session store using the Policy Server Management Console.

  For information about the session store, see the *Policy Server Administration Guide*.

- Sign-out requires a valid SiteMinder persistent session. which is established during Single Sign-on. Configure persistent sessions for the realm with the protected resources, including the authentication URL, at the Resource Partner.

  For information about realms, see the *Policy Server Configuration Guide*.

Follow these steps:

1. Log in to the Administrative UI.
2. Select the WS-Federation partnership that you want to modify.
3. Navigate to the Single Sign-on and Sign-Out step of the partnership wizard.
4. In the Sign-Out section, set the following fields:
   - Enable Sign Out
   - Sign-Out Confirm URL (IP only)
   - Sign-Out URL

   The URLs must each have an entry that starts with https:// or http://.
5. Navigate to the Confirm step and click Finish to save your changes.

Sign-out is configured.

Local Logout at the SP (SAML 2.0)

CA SiteMinder as an SP supports local logout for stand-alone applications. Local logout enables a user to be logged out at the local SP-side application. The session at the SP is removed, but no communication with the IdP or other SPs is involved. Sessions at the IdP and other SPs remain active.

If you include a logout link in an application at the SP, the SP sends a logout request to the local single logout service. The SP logs out the user upon receiving the request. The application at the SP is responsible for sending a confirmation message that the logout is successful.

CA SiteMinder provides local logout using a query parameter named **localLogout**. To use this parameter, your application can have a page, such as the following example:

```
You have completed your registration with demoapp.
To end your session securely, select LOGOUT.
```

The following sample string represents the link for the LOGOUT button:

```
<http://sp1server.demo.com:8080/affwebservices/public/saml2slo?LocalLogout=true
```

Chapter 19: Authentication Context Processing (SAML 2.0)

The *authentication context* indicates how a user authenticated at an Identity Provider. The Identity Provider includes the authentication context in a single sign-on assertion at the request of a Service Provider or based on configuration at the Identity Provider. A Service Provider can require information about the authentication process to establish a level of confidence in the assertion before granting access to resources.

Requesting the Authentication Context

To request the authentication context, the CA SiteMinder Service Provider must include the <RequestedAuthnContext> element in the authentication request to the Identity Provider. The Service Provider, puts this element is in the request based on a configuration setting in the SP->IdP partnership.

Obtaining the Authentication Context

A CA SiteMinder Identity Provider obtains the authentication context in *one* of two ways:

- You specify a static AuthnContext URI in the IdP->SP partnership configuration.

  If the federated partner is a CA SiteMinder Service Provider that does not support AuthnContext requests, manually enter a URI in the Administrative UI.

- The AuthnContext URI is determined dynamically using a configured authentication context template.

  The Policy Server maps the authentication context URIs to Policy Server-defined authentication levels. The authentication levels indicate the strength of an authentication context for an established user session. The levels enable the authentication context to be derived from the user session at the Identity Provider.

  When the Identity Provider receives a request, it compares the value of the <RequestedAuthnContext> element to the authentication context. The comparison is based on a comparison value in the request from the Service Provider. If the comparison is successful, the Identity Provider includes the authentication contexts in the assertion that it returns to the Service Provider. If validation is configured at the Service Provider, the Service Provider validates the incoming authentication context with the value it requested.

Authentication Context Processing for IdP-initiated SSO

When single sign-on is initiated at the IdP, authentication context processing follows these steps:

1. A user request triggers single sign-on at the IdP.
2. The user is authenticated and a user session is generated. Associated with the session is a protection level that is configured with the authentication scheme.
3. Depending on the authentication context configuration at the IdP, *one* of the following conditions occur:
   - Automatic detection occurs

     Based on a configured authentication context template, the AuthnContext class is mapped to the protection level for the session.
   - Predefined authentication class is used.

     The hard-coded URI you specify is added to the assertion.
4. The IdP generates the assertion and adds the authentication context to it. The assertion is then sent to the SP.
5. At the SP, another comparison is made between the authentication context class from the assertion and the one configured at the SP. If this comparison is successful, the authentication transaction is complete.

Authentication Context Processing for SP-Initiated SSO

When single sign-on is initiated at the SP, authentication context processing follows these steps:

1. The SP sends an authentication request with the <RequestedAuthnContext> element and a comparison operator. The element is included based on a setting in the configuration of the SP-> IdP partnership.
2. When the IdP receives the request, the IdP authenticates the user and a user session is generated. Associated with the session is a protection level for the authentication scheme.
3. Depending on the authentication context configuration at the IdP, *one* of the following conditions occur:
   - Automatic detection occurs

     Based on a configured authentication context template, the AuthnContext class is mapped to the protection level for the session.
   - Predefined authentication class is used

     The hard-coded URI you specify is added to the assertion.

4. The IdP compares the AuthnContext against the authentication class for the user session. The comparison is based on the comparison operator that is sent with the request. See the table that follows this procedure for examples of how each comparison operator affects processing.

   If the SP includes multiple authentication context URIs in the request, the classes are compared one-by-one in sequential order against the context for the session. At the first successful comparison, the IdP adds the session authentication context to the assertion.

5. If the comparison is successful, then the authentication context is added to the assertion sent to the SP.

   If the comparison is not successful, the transaction is terminated with a "noauthncontext" status response.

6. At the SP, a second comparison takes place between the authentication context from the assertion and the one configured at the SP. If this comparison is successful, the authentication transaction is complete.

The following table shows examples of how an authentication context is processed depending on the comparison attribute sent in the authentication context request.

SP-requested Authentication Context	Comparison Attribute Value	IdP-configured Authentication Context	Status Response
Password	exact	InternetProtocol	NoAuthnContext
Password	minimum	InternetProtocol	NoAuthnContext
Password	better	InternetProtocol	NoAuthnContext
InternetProtocol	exact	InternetProtocol	Success
InternetProtocol	minimum	InternetProtocol	Success
InternetProtocol	maximum	InternetProtocol	Success
InternetProtocol	maximum	Password	NoAuthnContext
InternetProtocol	better	Password	Success

Authentication Context Template Overview

An authentication context template defines the specific SAML 2.0 AuthnContext URIs that a partner supports. Each URI identifies a particular context class is aassigned a protection level and the protection level is then mapped to a strength level.

You can select a template on a per-partnership basis; multiple partnerships can use a single template.

A template has the following distinct functions at each partner:

At the IdP

An authentication context template is required at the IdP when the IdP is configured to automatically detect the authentication context from the SP request.

The template maps URIs to the protection levels associated with a user session. The protection levels indicate the strength of the authentication scheme at the policy server, from 1 through 1000, with 1000 being the strongest. An administrator assigns protection levels when configuring an authentication scheme that authenticates a user and establishes a user session.

The IdP first uses the template to determine the strength of the user session. It then uses the template to determine the strength of the URI in the SP authentication request. These strength levels are then compared.

At the SP

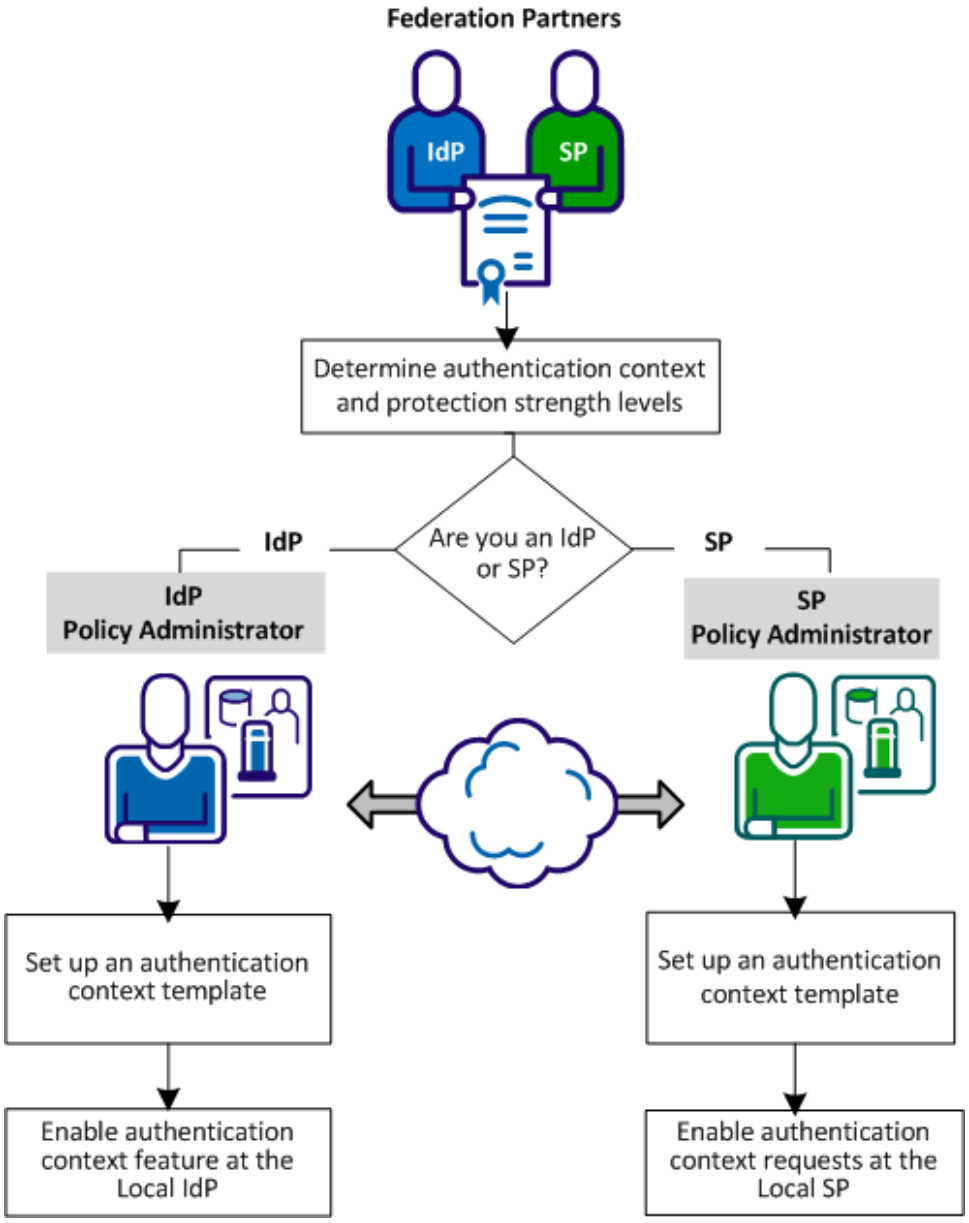
An authentication context template at the SP is required to generate an authentication context that is sent in the authentication request. After the SP generates the request, it sends it to the IdP. The template is also required for the SP to validate that the received assertion satisfies the authentication context requested.

Before proceeding with configuration, verify that you meet the following minimum knowledge requirements:

- Familiarity with SAML 2.0 standards related to authentication context processing.
- An understanding of federation configuration objects.
- Knowledge of how to access and use the Administrative UI.

Authentication Context Template Configuration

-The following figure shows the configuration process for each partner. CA SiteMinder Federation does not have to be installed at each site.



Complete the following steps to configure authentication context processing:

1. Determine authentication context and strength levels with your partner.
2. Set up an authentication context template.
3. Complete the task for your site:
   - Enable authentication context processing at the local IdP partnership.
   - Enable authentication context requests at the local SP partnership.

Determine Authentication Context and Strength Levels with your Partner

The SP can require specific authentication contexts and strength levels before it permits access to a requested resource. Based on the sensitivity of the resources at the SP, the SP has to have confidence in the assertion it receives from the IdP.

The administrators at the IdP and SP have to establish guidelines for supported authentication contexts and the relative strength of each authentication context URI. The order of the URIs at the IdP together with the associated strength levels affects how the IdP responds to the SP.

For example, an SP requests an authentication context for an X.509 certificate and a comparison value of exact. The IdP has to authenticate the requesting user at a suitable strength level and satisfy the comparison value during the evaluation of the authentication context.

Set up an Authentication Context Template

Set up an authentication context template to implement authentication context processing. This procedure is the same for an Identity Provider or Service Provider.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Authentication Context Templates.

   The View Authentication Context Templates window opens.
3. Select Create Template.

   The template wizard opens at the first step.
4. Enter a name for the template.
5. Complete *one* of the following actions:
   - Manually enter a URI and click Add URI.
   - Click Load Default URIs to select URIs from a predefined list. Move URIs from the Available URIs to the Selected URIs list.

6. Arrange the selected URIs by strength level. The strength level is in descending order, with the strongest URI at the top and the least strong at the bottom.
7. Click Next.
8. (Optional) Group URIs that require the same level of strength indenting one URI under the previous URI. Use the Change Grouping arrow to move a URI into or out of a group.
9. Click Enable Protection Levels.

   Map the protection levels from an authentication scheme to the URIs. The protection levels indicate the strength of an authentication scheme, ranging between 1 through 1000, with 1000 being the strongest. Individual URIs can have unique protection levels; however, grouping URIs means that they have the same level of strength.

   Consider the following information when assigning protection levels:

   - Assign the protection levels in descending order. List the strongest context at the top and the weakest context at the bottom.
   - You can modify the maximum protection level and the Administrative UI calculates the minimum. The Administrative UI verifies that there is no gap in the range of levels so that each protection level has an associated URI.

   Read more about protection level assignments.
10. Select Finish to confirm the configuration.

The template is complete.

Protection Level Assignments for a Context Template

The protection levels indicate the strength of the authentication. Assign protection levels to each selected authentication context URI. Specify the maximum level for each URI in the list. The minimum protection level is automatically determined based on the maximum level for the subsequent URI in the list. This range reflects the protection level.

The protection level assignments must reflect the protection levels of the configured Policy Server authentication schemes. For example, the Policy Server can have an X.509 authentication scheme at a protection level of 20. The protection level range that the template specifies must include 20. Finally, the Policy Server generates a URI strength level that is based on the protection level.

Example

Authentication Schemes set at the Policy Server	Protection Level
urn:oasis:names:tc:SAML:2.0:ac:classes:X509	20

Authentication Schemes set at the Policy Server	Protection Level
urn:oasis:names:tc:SAML:2.0:ac:classes:X509	20
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract	15
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol	10
urn:oasis:names:tc:SAML:2.0:ac:classes:Password	5

For each URI, the Policy Server automatically maps the protection level to a URI strength level. The ranges cover the protection level of the authentication scheme. For example:

- X509 scheme covers protection levels 16-1000
- MobileTwoFactorContract covers protection levels 11-15
- Internet Protocol covers 6-10
- Password covers 1-5

URI	Protection Level Max	URI Strength
urn:oasis:names:tc:SAML:2.0:ac:classes:X509	1000	4
urn:oasis:names:tc:SAML:2.0:ac:classes: MobileTwoFactorContract	15	3
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol	10	2
urn:oasis:names:tc:SAML:2.0:ac:classes:Password	5	1

If you group several of the URIs, the grouping enables URIs with different protection levels to have the same URI strength. This strength means that the URIs are considered equivalent.

The following modified table shows the grouping of the X.509 URI and the MobileTwoFactorContract URI.

URI	Protection Level Max	URI Strength
urn:oasis:names:tc:SAML:2.0:ac:classes:X509	1000	3
urn:oasis:names:tc:SAML:2.0:ac:classes: MobileTwoFactorContract	800	3
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol	700	2
urn:oasis:names:tc:SAML:2.0:ac:classes:Password	200	1

The range of strength levels reflects the total number of groups in the list. For example, if there are three groups, the strength level ranges from 1 to the total number groups, which are 3.

Enable Authentication Context Processing at the Local IdP Partnership

The Policy Server acting as the IdP can obtain the authentication context for an assertion in these two ways:

- Use a predefined authentication class

  Specify a URI for the authentication class and ignore the context request from the SP. A hard-coded entry can act as the default authentication context for IdP-initiated single sign-on.

- Detect the authentication class automatically.

  The Policy Server automatically detects the user session authentication context using the authentication context template.

  The IdP uses the template even if the authentication request from the SP does not include the <RequestedAuthnContext> element. The presence of the element triggers extra evaluation by the IdP and constrains the choices of what it can put in the assertion.

  You can find more information about the flow of authentication context processing (see page 212).

Follow these steps:

1. Navigate to the SSO and SLO step in the IdP->SP partnership wizard.
2. In the Authentication section, specify how to obtain the authentication context. Use a predefined authentication class or an automatically detected class with an authentication context template.
3. Follow the steps for the method chosen in the previous step:
   - To include a predefined class in the assertion, select a URI from the Authentication Class pull-down menu.
   - To include a class from the session context and a template, select a template from the authentication Context Template field or click Create Template.
4. (Optional). Depending on how you obtain the authentication context you can also select the Ignore RequestedAuthnContext check box.

The following table shows how the Configure AuthnContext and the Ignore RequestedAuthnContext settings work together:

Configure AuthnContext	Ignore RequestedAuthnContext	SP requests AuthnContext	Result
Predefined Class	Selected	Yes	IdP ignores the <RequestedAuthnContext> and uses the defined value in the assertion.
Predefined Class	Selected	No	IdP returns the defined value in the assertion by default.
Predefined Class	Not selected	Yes	Transaction fails because the IdP is not configured to handle the authentication context request. The IdP returns an error message to the SP.
Predefined Class	Not selected	No	IdP returns the defined class value in the assertion by default.
Automatically Detect Class	Selected	Yes	IdP compares the protection level for the authentication scheme against the authentication context template and returns the matching authentication URI in the assertion. The IdP ignores the values in the SP request.
Automatically Detect Class	Selected	No	IdP compares the protection level for the authentication scheme against the authentication context template and returns the matching authentication URI in the assertion. The IdP ignores the values in the SP request.
Automatically Detect Class	Not selected	Yes	IdP compares the protection level against the authentication context class that the SP sends. The IdP uses the authentication context template to determine the authentication URI it places in the assertion.

Configure AuthnContext	Ignore RequestedAuthnContext	SP requests AuthnContext	Result
Automatically Detect Class	Not selected	No	IdP compares the protection level for the authentication scheme against the authentication context template and returns the matching authentication URI in the assertion.

Enable Authentication Context Requests at the Local SP Partnership

The authentication context is part of an assertion authentication statement and it indicates how a user authenticated at an IdP. An SP can require information about the authentication process to establish a level of confidence in the assertion before granting access to resources.

Authentication Context URIs are the value of the <AuthnContextClassRef> element inside of a <AuthnContext> element. Each URI identifies the context class that the SP wants the IdP to return in the assertion.

The authentication context template at the SP defines the following information:

- Which URIs the SP wants to receive from the IdP. For outgoing requests, the URIs in the template indicate which authentication contexts are acceptable to the SP before it allows access to the requested resource.
- How the URIs in the request are compared to the URIs defined at the IdP.
- How the SP uses the URIs. The SP can include URIs in the outgoing authentication request. The SP can also validate URIs in the incoming assertion response. You can configure the URI usage for both functions.

You can select a template on a per-partnership basis and multiple partnerships can use a single template.

Configure an authentication context template before you enable authentication context requests or while you are configuring the SP partnership.

Follow these steps:

1. Log in to the Administrative UI.
2. Select the SP->IdP partnership you want to edit.

3. Navigate to the Configure AuthnContext step in the partnership wizard.

   The configuration dialog opens.

4. Select the Enable Authentication Context Processing check box.

5. Complete the fields in the dialog. Click Help for a description of fields, controls, and their respective requirements.

   Note the following information:

   - If no authentication context template exists, select Create template.
   - The Comparison field describes how the URIs in the SP authentication request are compared with the URIs configured at the Identity Provider.

     The Help details each comparison operator.
   - If you are selecting URIs from the Available URIs list, the available URIs reflect the URIs configured for the chosen template. If there are no predefined templates, click Create Template to configure one.

The authentication context request is included in the authentication requests sent to the Identity Provider.

Chapter 20: Sign and Encrypt Federation Messages

This section contains the following topics:

Key and Certificate Management for Federation

Securing an assertion and encrypting data within the assertion is a critical part of partnership configuration. In a federation environment, key/certificate pairs and standalone certificates serve a number of functions:

- Signing/verification of assertions (all three profiles)
- Signing/verification of authentication requests (SAML 2.0 only)
- Signing/verification of single logout requests and responses (SAML 2.0)
- Signing back channel requests and responses for HTTP-Artifact SSO (SAML 1.1 and 2.0)
- Encryption/decryption of an entire assertion or part of an assertion (SAML 2.0)
- Client credentials across the back channel for artifact single sign–on (SAML 1.1 and 2.0)

The *Policy Server Configuration Guide* contains information and instructions about managing keys and certificates.

You can use SSL server certificates to do the following tasks:

- Manage federation traffic across an SSL connection.
- Secure communication across the back channel for artifact single sign-on.

Refer to instructions for enabling SSL for the web server where the CA SiteMinder Web Agent is installed.

Note: If you enable SSL, it affects all URLs for all services, even the Base URL parameter. This means that all service URLs must begin with https://.

SAML 2.0 Signing Algorithms

For SAML 2.0, you have the option of choosing a signing algorithm for signing tasks. The ability to select an algorithm supports the following use cases:

- An IdP-->SP partnership in which the IdP signs assertions, responses and SLO-SOAP messages with the RSAwithSHA1, or the RSAwithSHA256 algorithm.
- An SP-->IdP partnership in which the SP signs authentication requests and SLO-SOAP messages with the RSAwithSHA1, or the RSAwithSHA256 algorithm.

Signature verification automatically detects which algorithm is in use on a signed document then verifies it. No configuration for signature verification is required.

Signature Configuration at a SAML 1.1 Producer and WSFED IP

The Signature step lets you define how the Policy Server uses private keys and certificates to sign SAML assertion or WS-Federation token responses. For SAML 1.1, you can elect to sign only assertions instead of the assertion response.

SAML 1.1 and WS-Federation do not support encryption.

There can be multiple private keys and certificates in the certificate data store. If you have multiple federated partners, you can use a different key pair for each partner.

Note: If the system is operating in FIPS_COMPAT or FIPS_MIGRATE mode, all certificate and key entries are available from the pull-down list. If the system is operating in FIPS-Only mode, only FIPS-approved certificate and key entries are available.

Follow these steps:

1. Log in to the Administrative UI
2. Select the asserting-to-relying party partnership that you want to modify.
3. Navigate to the Signature step in the partnership wizard.
4. In the Signature section, select an alias from the pull-down list for the Signing Private Key Alias field.

   If there is no private key in the certificate data store, click Import to import a key. Alternatively, click Generate to create a certificate request.

   By completing this field, you are indicating which private key the asserting party uses to sign assertions and responses.

5. (SAML 1.1 only) For the Artifact and Post signature options, select the specific components (assertion, response) that you want signed.

If you are using CA SiteMinder in a test environment, you can disable signature processing to simplify testing. Click the Disable Signature Processing check box.

Signature configuration is complete.

Signature Verification at a SAML 1.1 Consumer and a WSFED RP

The Signature step lets you define how the Policy Server uses private keys and certificates to verify SAML assertion or WS-Federation token responses. For SAML 1.1, you can elect to verify only assertions.

SAML 1.1 and WS-Federation do not support encryption.

There can be multiple private keys and certificates in the certificate data store. If you have multiple federated partners, you can use a different key pair for each partner.

Note: If the system is operating in FIPS_COMPAT or FIPS_MIGRATE mode, all certificate and key entries are available from the pull-down list. If the system is operating in FIPS-Only mode, only FIPS-approved certificate and key entries are available.

Follow these steps:

1. Log in to the Administrative UI
2. Select the relying-to-asserting party partnership that you want to modify.
3. Navigate to the Signature step in the partnership wizard.
4. Select an alias from the certificate data store for the Verification Certificate Alias field.

   By completing this field, you are indicating which certificate verifies signed assertions or responses or both. If there is no certificate in the certificate data store, click Import to import one. Alternatively, click Generate to create a certificate request.

Note: If you are using the product in a test environment, you can disable signature processing to simplify testing. Click the Disable Signature Processing check box.

Signature configuration is complete.

Signature Configuration at a SAML 2.0 IdP

The Signature and Encryption step in the partnership wizard lets you define how the product uses private keys and certificates for the following signing functions:

- Sign and verify SAML assertions, assertion responses, and authentication requests.

  For SAML 2.0 POST binding, you are required to sign assertions.

- Sign single logout responses and requests (HTTP-Redirect and SOAP bindings).

There can be multiple private keys and certificates in the certificate data store. If you have multiple federated partners, you can use a different key pair for each partner.

Note: If the system is operating in FIPS_COMPAT or FIPS_MIGRATE mode, all certificate and key entries are available from the pull-down list. If the system is operating in FIPS-Only mode, only FIPS-approved certificate and key entries are available.

To configure signing options

1. Select the Signature and Encryption step in the partnership wizard.
2. In the Signature section, select an alias for the Signing Private Key Alias field. If there is no private key available, click Import to import one. Or, click Generate to create a certificate request.

   By completing this field, you are indicating which private key the asserting party uses to sign assertions, single logout requests and responses.

   Note: click on Help for a description of the fields.

3. Select the hash algorithm for digital signing in the Signing Algorithm field. The IdP signs assertions, responses and SLO-SOAP messages with the specified algorithm.

   Select the algorithm that best suits your application.

   RSAwithSHA256 is more secure than RSAwithSHA1 due to the greater number of bits used in the resulting cryptographic hash value.

   The system uses the algorithm that you select for all signing functions.

4. Select an alias from the certificate data store or the Verification Certificate Alias field.

   By completing this field, you are indicating which certificate verifies signed authentication requests or single logout requests or responses. If there is no certificate in the database, click Import to import one.

5. (Optional) Specify Artifact and POST signature options for the assertion or response or both.
6. (Optional) Specify an SLO SOAP signature option for the logout request, the logout response or both when you are using single logout.

7. (Optional) Select the check box for Require Signed Authentication Requests. This check box verifies that the asserting party only accepts signed requests from the relying party.

   Activate a partnership for all configuration changes to take effect and for the partnership to become available for use. Restarting the services is not sufficient.

If you are using the product in a test environment, you can disable signature processing to simplify testing. Click the Disable Signature Processing check box.

Important! Enable signature processing in a SAML 2.0 production environment.

Encryption Configuration at a SAML 2.0 IdP

The Signature and Encryption step in the Partnership wizard lets you define how the Policy Server uses private keys and certificates to do the following tasks:

- Sign and verify SAML assertions, assertion responses, and authentication requests.

  For SAML 2.0 POST binding, you are required to sign assertions.
- Sign single logout responses and requests (HTTP-Redirect and SOAP bindings).
- Encrypt and decrypt entire assertions, Name IDs and attributes.

There can be multiple private keys and certificates in the certificate data store. If you have multiple federated partners, you can use a different key pair for each partner.

To configure encryption options

1. In the Encryption section, select one or both of the following check boxes to specify the assertion data to be encrypted:
   - Encrypt Name ID
   - Encrypt Assertion
2. Select the certificate alias from the certificate data store for the Encryption Certificate Alias.

   This certificate encrypts assertion data. If no certificate is available, click Import to import one.
3. Select values for the Encryption Block Algorithm and Encryption Key Algorithm fields.

   For the following block/key algorithm combinations, the minimum key size that is required for the certificate is 1024 bits.
   - Encryption Block Algorithm: 3DES

     Encryption Key Algorithm: RSA-OEAP

- Encryption Block Algorithm: AES-256

  Encryption Key Algorithm: RSA-OEAP

  Note: To use the AES-256 bit encryption block algorithm, install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files. You can download these files from http://java.sun.com/javase/downloads/index.jsp.

The encryption configuration is complete.

Signature Configuration at a SAML 2.0 SP

The Signature and Encryption step in the partnership wizard lets you define how the Policy Server uses private keys and certificates to do the following tasks:

- Verify SAML assertions signatures and assertion responses and sign authentication requests.

  Note: For SAML 2.0 POST binding, the IdP is required to sign assertions.

- Sign single logout responses and requests (HTTP-Redirect and SOAP bindings).

There can be multiple private keys and certificates in the certificate data store. If you have multiple federated partners, you can use a different key pair for each partner.

Note: If the system is operating in FIPS_COMPAT or FIPS_MIGRATE mode, all certificate and key entries are available from the pull-down list. If the system is operating in FIPS-Only mode, only FIPS-approved certificate and key entries are available.

To configure signing options

1. Begin by selecting the Signature and Encryption step in the partnership wizard.
2. In the Signature section, select an alias from the certificate data store for the Signing Private Key Alias field. If there is no private key in the database, click Import to import one. Or, click Generate to create a key pair and generate a certificate request.

   By completing this field, you are indicating which private key the relying party uses to sign authentication requests and single logout requests and responses.

   Note: Click Help for a description of fields, controls, and their respective requirements.

3. Select the hash algorithm for digital signing in the Signing Algorithm field. The SP signs authentication requests and SLO-SOAP messages with the specified algorithm.

   Select the algorithm that best suits your application.

   RSAwithSHA256 is more secure than RSAwithSHA1 due to the greater number of bits used in the resulting cryptographic hash value.

   CA SiteMinder uses the algorithm that you select for all signing functions.

4. Select an alias from the certificate data store for the Verification Certificate Alias field.

   By completing this field, you are indicating which certificate the relying party uses to verify signed assertions or single logout requests and responses. If there is no certificate in the database, click Import to import one.

5. (Optional) For the SP to sign all authentication requests, select the Sign Authentication Requests. If the remote asserting party requires the authentication requests to be signed, check this option.

   Activate a partnership for all configuration changes to take effect and for the partnership to become available for use. Restarting the services is not sufficient.

If you are using CA SiteMinder in a test environment, you can disable signature processing to simplify testing. Click the Disable Signature Processing check box to disable the feature.

Important! Enable signature processing in a SAML 2.0 production environment.

Encryption Configuration at a SAML 2.0 SP

The Signature and Encryption step lets you configure how the SP uses private keys and certificates, including encrypting and decrypting assertions, Name IDs, and attributes.

There can be multiple private keys and certificates in the certificate data store. If you have multiple federated partners, you can use a different key pair for each partner.

Note: If the system is operating in FIPS_COMPAT or FIPS_MIGRATE mode, all certificate and key entries are available from the pull-down list. If the system is operating in FIPS-Only mode, only FIPS-approved certificate and key entries are available.

To configure encryption options

1. In the Encryption section, select one or both of the following check boxes so that the correct data is encrypted in the assertion:

   - Require encrypted Name ID
   - Require encrypted Assertion

   Note: To use the AES-256 bit encryption block algorithm, install the Sun Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files. You can download these files from http://java.sun.com/javase/downloads/index.jsp.

2. Select the alias from the certificate data store for the Decryption Private Key Alias.

   This private key decrypts any encrypted assertion data. If no certificate available, click Import to import one or click Generate to create a key pair and generate a certificate request.

The encryption configuration is complete.

Chapter 21: Secure a Federated Environment

This section contains the following topics:

Methods to Secure Federated Transactions

Several mechanisms help secure transactions between federated partners, such as encrypting assertions and using SSL connections between partner sites.

When setting up a federated environment with partnership federation, here are some recommendations for protecting your environment:

- Generating assertions for only one time use.
- Protecting against cross-site scripting.

These topics are described in the following sections.

Enforcing the One Time Use of an Assertion

Reusing an assertion beyond its validity results in authentication decisions from out-of-date identity information. To prevent reuse, CA SiteMinder can generate an assertion for one-time use, in compliance with the SAML 1.x and 2.0 specifications. The assertion contains elements that tell the relying party not to retain the assertion for future transactions, preventing problems from reusing an assertion.

If CA SiteMinder is acting as the asserting party (Producer/IdP), you can configure the one time use of an assertion. For a SAML 1.x producer, you can select the **Set DoNotCache Condition** setting. For a SAML 2.0 IdP, you can select the **Set OneTimeUse Condition** setting. Both of these configuration settings enable CA SiteMinder to insert the proper elements in an assertion that indicate the one-time use condition.

Note: Do not confuse the one time use of an assertion with the single use policy for SAML 1.x and 2.0 HTTP-POST single sign-on. CA SiteMinder uses the single use policy when acting as the relying party, and it is only for POST transactions. The one time use feature is for HTTP-Artifact and HTTP-POST.

Securing Connections Across the Federated Environment

Identity information that is sent between federated partners or a partner and an application is best protected when communication takes place over a secure connection.

Securing the Connection Between the Relying Party and the Target Application

Secure data transmission from the relying party to the client-site target application. Using a secure connection as the communication channel makes your environment less vulnerable to security attacks.

For example, an assertion can contain attributes that the relying party extracts and sends to the client application. The relying party can pass these attributes to the application using HTTP header variables or cookies. Attributes stored in headers or cookies can be overwritten at the client side, allowing a malicious user to impersonate other users. Using an SSL connection protects an environment from this type of security breach.

As a best practice, protect against this vulnerability by setting the UseSecureCookies parameter in the appropriate Agent Configuration Object (ACO). The UseSecureCookies parameter instructs Federation Web Services to generate cookies that are marked with the "secure" flag. This flag indicates that the cookie is sent only over an SSL communication channel.

Note: The ACO to modify differs depending on the setup of your federation environment. If you deploy Federation Web Services on the same system as the Web Agent is installed, edit the ACO for the Web Agent. If you deploy Federation Web Services on a different system than the Web Agent, edit the unique ACO you created for Federation Web Services.

Securing the Initial Authentication at the CA SiteMinder Asserting Party

The initial authentication of a user at a CA SiteMinder asserting party presents a potential vulnerability. When a user first authenticates to establish a user session at the asserting party, a session ID cookie is written to the browser. If the cookie is sent over a non-SSL connection, an attacker can obtain the cookie and can steal sensitive user information. The attacker can then use the information, for impersonation or identity theft.

As a best practice, protect against this vulnerability by setting the Web Agent parameter UseSecureCookies, which you can modify in the Agent Configuration Object. The UseSecureCookies parameter instructs the Web Agent to generate cookies that are marked with the "secure" flag. This flag indicates that the browser passes the cookie only over an SSL connection, which increases security. In general, establishing SSL connections for all URLs is recommended.

Protecting a Federated Network Against Cross-Site Scripting

A Cross Site Scripting (XSS) attack can occur when an application displays input text from a browser. The application can possibly have failed to test for characters that can form an executable script. The display of these characters can lead to an unwanted script being executed on the browser.

CA SiteMinder provides several JSPs for use with federation functionality. These JSPs check characters in a request to be sure that unsafe information in the output stream is not displayed in the browser.

When CA SiteMinder receives a request, the following JSPs scan the decoded values for cross-site scripting characters:

- idpdiscovery.jsp

  Used at the relying party for Identity Provider Discovery.

- linkaccount.jsp

  Used at the relying party for dynamic account linking.

- sample_application.jsp

  Used at the IDP to initiate single sign-on. You can use this sample application to direct the user first to the SSO Service and then to the custom web application. Typically, you use your own application.

- signoutconfirmurl.jsp

  Used at the Account Partner for WS-Federation signout.

- unsolicited_application.jsp

  Used for IdP-initiated single sign-on when the user is sent directly to the web application and not initially to the SSO Service.

The pages scan the request for the following characters:

Character	Description
<	left-angle bracket
>	right-angle bracket
‘	single quotation mark
“	double quotation mark
%	percent sign
;	semi-colon
(	open (left) parenthesis

Character	Description
)	closed (right) parenthesis
&	ampersand
+	plus sign

Each JSP contains a variable that defines the characters to scan. Modify these JSPs to expand the character set.

Chapter 22: Application Integration at the Relying Party

This section contains the following topics:

Relying Party Interaction with Applications

The Application Integration step of the partnership wizard is applicable only at the relying party. This step lets you define various aspects of federated operation for resolving user identities and directing users to the target application.

The features that you can configure in the Application Integration step are:

- User redirection to the target application
- Mapping assertion attributes to application attributes (SAML only)
- Provisioning a user identity
- User redirection in case of an authentication failure

Redirecting a User to the Target Application

The Target Application section in the Application Integration step lets you define how a user gets redirected to the target application. The redirection method that you select depends on the type of data you want to pass with the user to the target application.

Follow these steps:

1. Navigate to the Application Integration step in the partnership wizard.
2. Select a redirection method for the Redirect Mode field. Note the following information:

   - If you select Cookie Data, you can URL-encode attribute data in the cookie by selecting the URL Encode Attribute Cookie Data check box. This option is only for SAML 1.1 and 2.0.
   - If you select the Open-format Cookie or the Open-format Cookie Post options, configure the additional required settings and optional settings. Unlike the Open-format Cookie, the Open-format Cookie Post sends the data in the form of an HTTP-POST request.

     If the relying party receives an assertion with multiple attribute values, the Policy Server passes all values to the target application in the cookie.
   - If you select one of the FIPS-compatible algorithms (AES algorithms), use a CA SiteMinder® Federation SDK to generate the open-format cookie. If you use the .NET SDK, use only the AES128/CBC/PKCS5Padding encryption algorithm.

     The target application must use the same language as the SDK that creates the cookie. If you are using the CA SiteMinder® Federation Java SDK, the application must be in Java. If you are using the .NET SDK, the application must support .NET.
   - If you select HTTP Headers as the redirect mode, CA SiteMinder can deliver multiple attribute values in a single header. Separate each attribute value with a comma. This option is only for SAML 1.1 and 2.0.

     Learn more about using HTTP Headers as the redirect mode (see page 237) and how to protect the headers.

   Click Help for a description of the fields.
3. Enter the URL of the target application in the Target field.

   If a proxy sits in front of the server with the target resource, enter the URL for the proxy host. The proxy handles all federation requests locally. The proxy host can be any system that sits in front of the target server. The proxy host can also be CA SiteMinder itself, provided it is being accessed directly from the Internet. Ultimately, when operating with a proxy, the URL you specify as the target must go through CA SiteMinder. For example, if the base URL is fed.demo.com and the back-end server resource is mytarget/target.jsp, the value for this field is http://fed.demo.com:5555/mytarget/target.jsp.

   For SAML 2.0, you can leave this field blank if you override it with the RelayState query parameter. The RelayState query parameter can part of the URL that triggers single sign-on. To enable this override, select the Relay state overrides target check box.

Setting up redirection to the target is complete.

Using HTTP Headers to Pass Assertion Data (SAML only)

For a SAML entity, the Policy Server can use HTTP headers to pass identity attributes from an assertion to a back-end application. A backend application can be a target application for single sign-on or a user provisioning application. The system passes these headers in an encrypted cookie.

The headers have the same name as the assertion attributes. For example, if the assertion attribute is "address", the application looks for the HTTP header "ADDRESS".

Assertion attributes are case-sensitive, but HTTP headers are not. The Policy Server cannot pass the same attributes that differ only by case sensitivity and then map them to HTTP headers. For example, the system cannot pass "address" and "Address" as headers at the same time. In general, do not use the attributes with the same names that are only different because of case sensitivity or format.

The following additional values are passed as headers:

- NAMEID
- FORMAT
- AUTHNCONTEXT

Protecting HTTP Headers

If an unauthorized user knows the name of an assertion attribute, that user can set this name as a header in a browser. With the header set, the malicious user can gain access to the target application. The target application sees an expected header value and grants access to the resource without CA SiteMinder consuming an assertion.

Setting a value for the FedHeaderPrefix protects against the following scenario:

1. An unauthorized user learns the names of HTTP headers. These header names include prefixes.
2. The malicious user sends an incoming request, including the headers, to the Policy Server.
3. The Policy Server recognizes that the headers containing prefixes come from an incoming request and are not generated internally so it removes these headers.
4. Before the system passes its own legitimate headers to the back-end application, it adds the specified prefix to each header. The headers are then passed to the application.

Configure HTTP Headers to Pass Assertion Data (SAML only)

CA SiteMinder can pass assertion data using HTTP headers.

Follow these steps:

1. Verify that the CA SiteMinder web agent is installed on the relying party system that is handling federation traffic.
2. Navigate to *web_agent_home*/conf and modify the WebAgent.conf file. Uncomment the following entry so it appears as follows:

   Windows

   LoadPlugin="*path*\SAMLDataPlugin.dll"

   UNIX

   LoadPlugin="*path*/SAMLDataPlugin.so"
3. (Optional but recommended) Add the setting **fedheaderprefix** setting to the appropriate Agent Configuration Object for the web agent. Enter any string as a prefix.

   The fedheaderprefix setting specifies a global prefix that CA SiteMinder adds to HTTP headers. Setting a prefix protects HTTP headers against manipulation by an unauthorized user before the CA SiteMinder consumes an assertion. As a result, only legitimate headers get passed to the target application. Read more about protecting HTTP headers (see page 238).
4. Do *one* of the following tasks in the Application Integration step of the partnership wizard:
   - Select HTTP Headers as the Redirect Mode for the target application.
   - Select HTTP Headers as the Delivery Option for user provisioning.

HTTP headers are now configured to pass attribute data.

Mapping Assertion Attributes to Application Attributes (SAML only)

At a SAML 1.1 consumer or SAML 2.0 SP, you can map a set of assertion attributes to a set of outgoing application attributes. The application attributes are then delivered to the target application. Attribute mapping allows you to provide a customized experience for users without having to modify the target application. Attributes are mapped on a per-partnership basis, which allows you to use a relying party-side application for multiple asserting parties.

The following types of mapping are available:

- Convert assertion attribute names to application attribute names.

  Example

  An incoming assertion attribute can be Region=US. The attribute can be converted to an outgoing application attribute ServiceLocation=US.

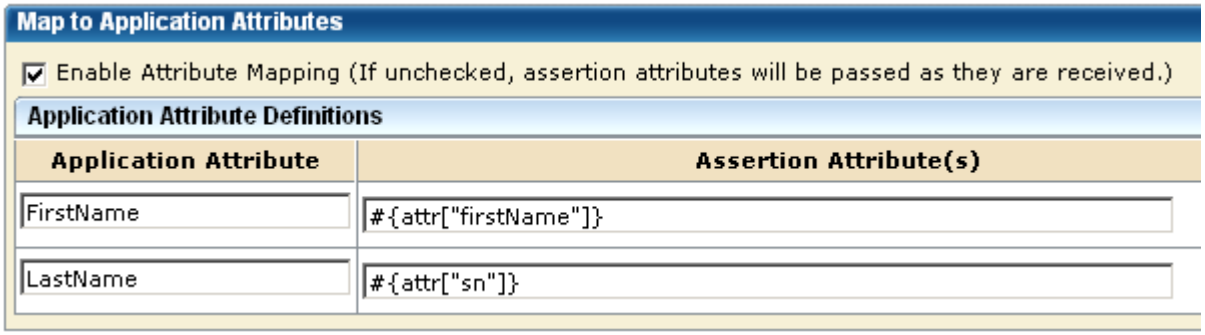
- Transform separate attributes and their values into a single attribute.

  Example

  Two attributes are included in the assertion, Name=Bob and LastName=Smith. These two attributes can be converted to FullName =Bob Smith.

Using the Application Attributes Definitions Table

You define attribute mapping rules in the Application Attributes Definitions table of the Application Integration dialog. This table is shown in the following figure:



The Application Attribute and Assertion Attribute(s) columns are populated using assertion attributes for the remote Producer or IdP entity. You configure these attributes at this local relying party. The assertion attribute name is entered for the Application Attribute column. The equivalent Unified Expression Language (UEL) string is entered in the Assertion Attribute(s) column.

Administrators or application integrators at the relying party must know the following information to configure attribute mapping:

- Names of the target application attributes.
- Names of the attributes in the assertion.
- Mapping relationship between the assertion attributes and the target application attributes. Understanding the mapping relationship means that you know how to transform the available assertion attributes into the required application attributes.

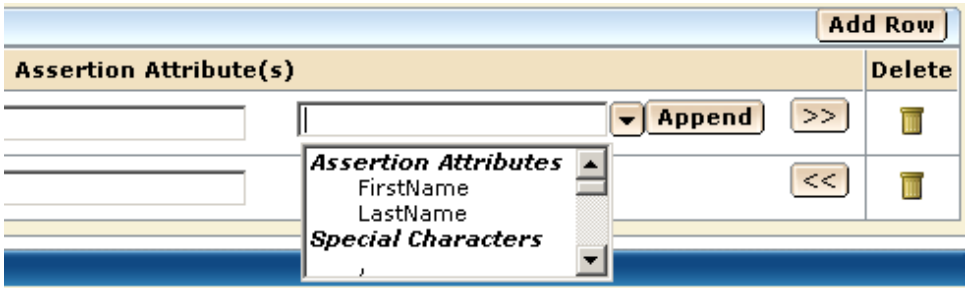
Gather the names of the application and assertion attributes from the necessary parties before setting up attribute mapping.

The application attributes must reflect the attributes that the target application uses so you must modify the default values to suit the application. You obtain the application attributes from an out-of-band communication with the application administrator.

Use the Expression Builder to Build Mapping Rules

The UI provides an expression builder to aid in the construction of mapping rules. Access the expression builder by selecting the slider button (<<) to the right of the Assertion Attribute(s) field. The slider button reveals a blank field and pull-down arrow. Select the arrow to see a list of assertion attributes and special characters that you can use to compose a mapping. Click the slider button (>>) to hide the expression builder.

The following figure shows the Expression Builder menu.



The Assertion Attributes list from the expression builder is populated from assertion attributes for the remote Producer or IdP entity. You configure these attributes at this local relying party. You can specify entries manually as long as you know that the attribute is in the assertion. You do not have to use only the options from the expression builder menu.

The Special Characters list contains characters, such as commas and percent signs that you can use to build a mapping rule. You can select a character from the list or you can enter the character manually.

Important! When you enter assertion attributes in this table, they are case-sensitive relative to the assertion attribute specified at the remote asserting party. The cases must match. If CA SiteMinder is at both sides of the partnership, the attributes are specified in the NameID and Attributes step of the remote IdP partnership wizard. Obtain the assertion attributes in an out-of-band communication with the partner or by importing metadata.

After the mapping rules are defined, CA SiteMinder places the data in a legacy cookie, an open format cookie, or an HTTP header. CA SiteMinder then sends the data to the application. You specify the delivery method in the Target Application section of the Application Integration dialog.

Modify and Delete Mappings

You can change or remove attribute mappings in the Application Attributes Definitions table at any time.

To modify a mapping

1. Place your cursor in any of the fields in the row you want to modify and enter the new text. You can also use the expression builder to append additional values to the end of the current expression.
2. Save the change by clicking Next to advance to the end of the wizard.

To delete a mapping

1. Click the trash barrel in the Delete column for the entry you want to remove.
2. Save the change by clicking Next to advance to the end of the wizard.

Construct Attribute Mapping Rules Using the Proper Syntax

Attribute mapping uses mapping rules that transform assertion attributes to application attributes. When you enable attribute mapping, CA SiteMinder generates default mapping rules. The rules are based on the assertion attributes specified for the remote Producer or IdP entity. All this configuration takes place at the local relying party. When you disable attribute mapping, assertion attributes are passed "as is" to the target application.

CA SiteMinder uses a Unified Expression Language (UEL) syntax for mapping that is similar to JSP and JSF. Each assertion attribute is put into a hashmap and assigned the **attr** keyword. A UEL expression evaluator goes through the list of mapping rules and applies them to the hashmap of assertion attributes. The expression evaluator then generates another hashmap containing the resulting application attributes. The hashmap of outgoing application attributes is converted into cookie contents or header variables and delivered to the target application.

To construct expressions, it is important to understand the syntax CA SiteMinder uses for the expressions.

Single Attribute Representation

To represent a single assertion attribute, use the following syntax:

#{attr["*attribute_name*"]}

Example: #{attr["Name"]} represents the value of the Name assertion attribute.

Composite Attribute Representation

Value expressions can be concatenated to form a composite value (with optional delimiter). To represent a composite assertion attribute, use the following syntax:

#{attr["*first_attribute*"]}*optional_character* #{attr["*second_attribute*"]}

Mapping Examples

The following examples are a series of mapping rules. These examples are presented in the following format:

application_attribute=assertion_attributes_expression

Name Example

Syntax

ID = #{attr["Name"]}

Sample Result

BobSmith

Simple Concatenation Examples

Syntax

FullName = #{attr["FirstName"]},#{attr["LastName"]}

Sample Result

Bob,Smith

Syntax

FullName = #{attr["LastName"]},#{attr["FirstName"]}

Sample Result

Smith,Bob

Spaces are considered special characters. If you want a space between attributes in an expression, enter a space. For example:

Syntax

FullName = #{attr["LastName"]}, #{attr["FirstName"]}

Sample Result

Smith, Bob

Date Examples

Syntax

Date = #{attr["month"]}/#{attr["dateOfMonth"]}/#{attr["year"]}

Sample Result

01/05/2010

Syntax

Date = #{attr["monthSymbol"]} #{attr["dateOfMonth"]}, #{attr["year"]}

Sample Result

January 5, 2012

Monetary Example

Syntax

Price = #{attr["amount"]}#{attr["currency"]}

Sample Result

2.50EUR

Email Address Examples

Syntax

EmailAddress = #{attr["userName"]}@#{attr["domainName"]}

Sample Result

JaneDoe@company.com

Syntax

AcmeEmailAddress = #{attr["AcmeIDKey"]}@acme.com

Sample Result

bsmith@acme.com

Configure Attribute Mapping at the Relying Party

Define a set of mapping rules that CA SiteMinder can apply to the assertion attributes. CA SiteMinder lets you map a specific assertion attribute or a combination of several application attributes. The result of the mapping can be a single application attribute or multiple attributes.

Follow these steps:

1. Navigate to the Application Integration step in the partnership wizard.
2. Select the Enable Attribute Mapping check box in the Map to Application Attribute section.

   An Application Attribute Definitions table displays.
3. Modify any existing application attribute or define new ones in the table. All application attributes are delivered to the target application.

   The syntax of the value in the Assertion Attribute column must comply with Unified Expression Language (UEL).

   Select the slider button (<<) to open the expression builder and display the options available to you. To add the item from the list to the attribute value, select the assertion or special character and click Append.

   Note: When you specify Cookie Data and any special character in the Application Attributes Table, select the URL Encode Attribute Cookie Data option. The check box is in the Target Application section of the dialog. Special characters can be added from the drop-down list or entered manually. Additionally, the target application must URL decodes the name and value of the application attribute received.
4. (Optional) If the default mappings are not sufficient, add as many rows as you like.

   By default, all assertion attributes defined at the remote Producer or IdP entity are included in the table with the default (straight) mappings. The original assertion attribute is not changed. You can modify these mappings.
5. Configure the method by which the application attributes are sent to the target application. You configure the method in the Target Application section of the Application Integration dialog.

Attribute mapping configuration is complete.

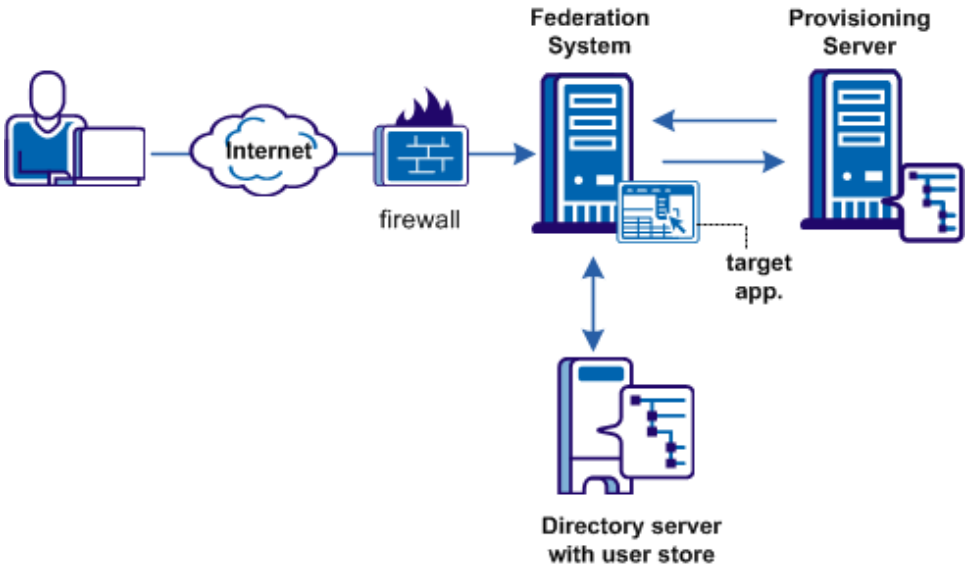
User Provisioning at the Relying Party

In a federated network, the relying party can establish accounts for users federating from different asserting parties. Dynamic provisioning supports the process of creating client accounts with the necessary account rights and access privileges for accessing data and applications.

Remote Provisioning

Remote provisioning employs a third-party provisioning application to create a user account. The application then passes the necessary information back to the Policy Server at the federation system with CA SiteMinder® Federation. The Policy Server uses the data to create a user credential.

Remote provisioning occurs at the relying party. The following figure shows a remote provisioning setup.



The high-level provisioning process is as follows:

1. The Policy Server at the relying party receives a request for a resource along with an assertion. However, the user cannot be found in the user directory.
2. With provisioning enabled, the Policy Server processes an active response containing assertion data and generates a cookie with the assertion data. Additionally, a cookie that keeps state is generated to indicate a provisioning request is in place.
3. The browser is redirected with an open-format cookie or headers to a provisioning application.
4. The provisioning application typically prompts the user to log in. After the user logs in, the application reads the cookie or the headers. The application uses the assertion data and the login credentials to establish a user account.

   The provisioning application can consume the open-format cookie using the CA SiteMinder® Federation Java or .NET SDK.

5. The browser redirects the user back to the assertion consumer service at the relying party after an account has been provisioned. A cookie that maintains state information about provisioning is examined to verify that the user has been provisioned. A credential is created and passed to the authentication scheme.

   Note: The provisioning application must know the URI of the assertion consumer service at the relying party. For example, the SAML 2.0 URI for CA SiteMinder as the relying party is https://*sp_server:port*/affwebservices/public/saml2assertionconsumer.

6. The Policy Server attempts user disambiguation a second time. Assuming that provisioning is successful, the user is authenticated and cookies or headers are sent to the target application.

   The redirect mode that you select for the target application determines the data delivery method to the target application.

7. The user is redirected to the target resource.

Delivery of Assertion Data to the Provisioning Application

To accomplish remote provisioning, CA SiteMinder redirects the browser with the assertion data to the provisioning application.

CA SiteMinder can pass the assertion data using one of these methods:

Open format cookie

Delivers SAML assertion information in an open-format cookie. The cookie contains a login ID based on the assertion data.

Note: If you use the open-format cookie, the CA SiteMinder system and the remote provisioning system must be in the same domain.

The cookie can be created in one of two ways:

- A CA SiteMinder® Federation SDK creates the cookie.

  If you select one of the FIPS algorithms (AES algorithms), use a CA SiteMinder® Federation SDK to generate the cookie. If you are planning to use the .NET SDK, use only the AES128/CBC/PKCS5Padding encryption algorithm. If the provisioning application uses .NET, the .NET SDK on the provisioning server reads the open format cookie.

  The provisioning application must use the same language as the SDK that it is using to create a cookie. If you are using the CA SiteMinder® Federation Java SDK, the application must be in Java. If you are using the .NET SDK, the application must support .NET.

- You manually create an open-format cookie.

  To create an open-format cookie without using a CA SiteMinder® Federation SDK, use any programming language. Review the details about the contents of the open-format cookie.

The language for writing the cookie must support UTF-8 encoding and any of the **PBE** encryption algorithms that you can select in the Administrative UI.

If you select FIPS-compatible (AES) algorithm to encrypt the cookie, the provisioning application must use an SDK to read the open-format cookie.

Verify that the open-format cookie gets set in the browser.

Open-format Cookie Post

The Open-format Cookie Post is similar to the open-format cookie, but it sends the data in the form of an HTTP-POST request. Use this option if you are concerned that data can be lost due to the cookie data limitations.

HTTP Headers

CA SiteMinder can also pass assertion information as HTTP headers. If you use HTTP headers, the CA SiteMinder system and the remote provisioning system can be in different domains.

Learn more about using HTTP headers to pass assertion data (see page 237) and how to protect the headers.

The delivery option is configurable in the Application Integration step of the partnership wizard.

After the user is redirected to the provisioning application, CA SiteMinder no longer has control over the process. If provisioning a user account is a time-consuming process, the provisioning application is responsible for handling this situation. For example, by the application can send a message to the user explaining that provisioning is in process. This information lets the user know not to keep trying to log in before a user account is available.

Remote Provisioning Configuration

To configure remote provisioning, determine a delivery option for the assertion data and supply the URL of the provisioning server.

In addition to configuring remote provisioning, you can select the Allow IdP to create User Identifier option. This option enables the IdP to create a persistent identifier if no identifier for the user exists. This Allow/Create feature is not exclusively for provisioning using local account linking, though it is required for the local method.

When you want the IdP to generate a user identifier that is sent with other attributes, you can enable the Allow/Create feature together with remote provisioning. The application at the remote provisioning server determines how it uses the generated identifier. The application can perform local account linking, but not CA SiteMinder local account linking.

To configure remote provisioning

1. Begin at the Application Integration step of the partnership wizard.
2. Select the provisioning type in the User Provisioning section.
3. If you select Remote as the provisioning type, complete the additional fields that are displayed.

   Click Help for a description of the fields.
4. Select the Confirm step and click Finish to save your changes.

You have completed remote provisioning configuration.

Failed Authentication Handling Using Redirect URLs (Relying Party)

Assertion-based authentication can fail at the site that consumes assertions. If authentication does fail, you can configure the Policy Server to redirect the user to different applications (URLs) for further processing. For example, when user disambiguation fails, you can configure CA SiteMinder to send the user to a provisioning system. Setting up redirect URLs is optional and is only configurable at the relying party.

Follow these steps:

1. Begin at the Application Integration step of the partnership wizard.

   In the Status Redirect URL section of the dialog, specify redirects only for the specific failure conditions that you want. For SAML 2.0, you can also configure redirects for specific HTTP error conditions.

   Click Help for a description of the fields.
2. For each redirect option you configure, specify the method by which CA SiteMinder redirects the user. The options are:

   302 No Data (default)

   Redirects the user with an HTTP 302 redirect and no data.

   HTTP Post

   Redirects the user with the HTTP Post protocol.

Configuration of the redirect URLs is complete.

Chapter 23: Export Metadata to Aid Partnership Configuration

This section contains the following topics:

Metadata Export Overview

A local entity generates metadata to help a remote entity create its entities and form partnerships. Metadata makes the partnership configuration more efficient because many aspects of the partnership are defined in the metadata file. A remote partner can import metadata and can create a partnership or a remote entity that is based on the information in a metadata document.

You can export metadata from an existing local asserting or relying entity.

The Administrative UI offers several options for exporting metadata:

- Export from a local entity.
- Export from a local partnership.
- Metadata exchange for local WSFED partnerships.

Regardless of whether you send metadata using a file or using the metadata exchange profile, the end goal of acquiring metadata is the same.

Note: For SAML 1.1, the terms in a metadata file are SAML 2.0 terms. This convention adheres to the SAML specification. When you import the SAML 1.1 data, the terms are imported correctly using SAML 1.1 terminology.

Entity-level Metadata Export

You can export data from a local entity. When you export metadata at the entity level, provide a partnership name for the data you are exporting. The export at this level defines basic partnership data.

Follow these steps:

1. Log in to the Administrative UI
2. Select Federation, Partnership Federation, Entities.
3. Click the Action pull-down menu next to any local entity in the list and select Export Metadata.

   The Export Metadata dialog opens.
4. Specify a new partnership name. The metadata file that results from the export contains information to establish a basic partnership.
5. Complete the remaining fields on the dialog. Be sure to fill in the settings in the Metadata Export Options section of the dialog.

   Note: Click Help for a description of fields.
6. Click Export.
7. A dialog prompting you to open or save the metadata file displays.

   Only open it to view it.
8. Save the data to an XML file on your local system.

The metadata is exported to the specified XML file. You can send this file to any partner.

Partnership-Level Metadata Export

You can export data from a local partnership. The export at this level defines basic partnership data.

Follow these steps:

1. Log in to the Administrative UI
2. Select Federation, Partnership Federation, Partnerships.
3. Select the Action pull-down menu next to any partnership in the list.
4. Select Export Metadata.

   The Export Metadata dialog opens.
5. Review the information. The metadata file that results from the export contains information to establish a basic partnership.

6. Complete the settings in the Metadata Export Options section for signing the metadata document and validating it.

   Note: Click Help for a description of fields, controls, and their respective requirements.

7. Click Export.
8. A dialog prompting you to open or save the metadata file displays.

   Only open it to view it.

9. Save the data to an XML file on your local system.

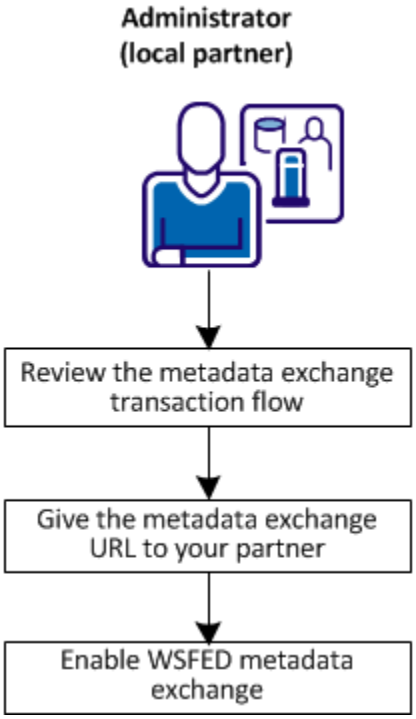
The metadata is exported to the specified XML file. You can send this file to any partner.

How To Enable WS-Federation Metadata Exchange

The Policy Server supports the Web Services Metadata Exchange profile for WS-Federation partnerships. This web service enables the CA SiteMinder local partner to respond to requests from a remote partner for metadata. The exchange occurs as an HTTP request and response.

The use of the HTTP protocol lets a remote entity configure the federation programmatically. An application can use the URL to gather the necessary information.

The following graphic shows the configuration steps for metadata exchange.



Complete the following configuration for metadata exchange:

1. Review the metadata exchange transaction flow. (see page 252)
2. Give the metadata exchange URL to your partner. (see page 252)
3. Enable WSFED metadata exchange (see page 253).

WS-Federation Metadata Exchange Supported for SAML 1.1

Important! WS-Federation Metadata Exchange is supported only for SAML 1.1.

Metadata Exchange Transaction Flow

A metadata exchange transaction has the following process flow:

1. A remote partner sends a request to the metadata exchange URL provided by the local partner.
2. The local partner sends the metadata back in an HTTP response to the remote partner. The Policy Server secures the metadata by signing the response. The certificate that lets the remote partner verify the response is in the response.

   The Policy Server generates the metadata document at the time of the request. This document is not stored at the local partner.
3. The remote partner verifies the signature of the response. Assuming the signature is valid, it parses the metadata document and uses the information to establish entities and partnerships.

Give the Metadata Exchange URL to Your Partner

Before any metadata transaction occurs, give the URL for metadata exchange requests to your remote partners. A federated partner must send the request to the following URL:

https://*server:port*/affwebservices/public/FederationMetadata/*partnership_name*

server:port

Name of the system hosting the metadata exchange service.

partnership_name

Name of a configured partnership.

Enable WSFED Metadata Exchange

Enable the metadata exchange feature at a local WS-Federation partner.

Follow these steps:

1. Log in to the Administrative UI.
2. Select the WSFED partnership that you want to modify.
3. In the Configure Partnership step of the partnership wizard, select the Enable Metadata Exchange check box.
4. Navigate to the Confirm step and click Finish.
5. Return to the main Partnership Federation tab (Federation, Partnership Federation).
6. Select Metadata Exchange Configuration in the left pane.

   The Metadata Exchange Configuration screen displays.
7. Provide the values to sign the response.
8. Click Save.

Metadata exchange is now configured for the partnership.

Chapter 24: Log Files that Aid Troubleshooting

This section contains the following topics:

Federation Trace Logging

The Federation Web Services (FWS) trace logging facility and the Policy Server Profiler monitor the performance of the federation services. These logging mechanisms provide information about federated operation so you can analyze the system performance and can troubleshoot issues.

Enable trace logging where the Web Agent Option Pack and the Policy Server are installed to extract in-depth information about federation processes. For example, you can look at the FWSTrace.log to see the generated SAML assertion or collect the name of the current user.

Note: Trace messages are ordinarily turned off during normal operation because they can impact performance.

The collected trace messages are written to two trace logs:

FWSTrace.log

The FWSTrace.log is located in the /log directory of the web server or application server where the web agent option pack is installed or deployed.

Web server

webagent/log

webagent_optionpack/log

Application server

default_deployment_directory/log

SPS federation gateway

sps_home/secure-proxy/proxy-engine/logs

smtracedefault.log

The smtracedefault.log is located in the directory *siteminder_home*/log.

siteminder_home represents the installation directory of the product.

In the FWSTrace.log and the smtracedefault.log, there are checkpoint log messages that indicate what is happening during a transaction. For example:

```
[07/30/2013][11:34:44][4260][5824][1181adbb-993f775c-33ba08f3-76b5
2f3b-3d2280cd-4ae][SSO.java][processRequest][Reading SAML 2.0 SP
Configuration [CHECKPOINT = SSOSAML2_SPCONFREAD_REQ]
```

You can search on these checkpoint messages to follow some of the processes occurring during a transaction.

In addition to the checkpoint messages, you can follow transaction IDs in the log to follow a transaction. If a transaction fails, the checkpoint messages and transaction IDs can help you determine the specific problem.

Transaction IDs to Aid Federation Troubleshooting

Troubleshooting a federated transaction is difficult when many transactions are logged in one file. To follow a single transaction in a trace log, use the SAML transaction ID. When a federation call occurs, the FWS application first generates a SAML Transaction ID. The SAML Transaction ID is generated only once. This unique SAML transaction ID can map to multiple transaction IDs

For example, you can see the following message in the fwstrace.log for a SAML 2.0 POST transaction. Note the line in bold that shows the mapping of the two transaction IDs.

```
[08/01/2013][17:33:54][2292][1884][1c2d7650-b006e46a-ed071f41-bbbe
de33-fe78e2dd-38d][SSO.java][processAuthentication]
```
[SAMLTransactionID 2aaf90ec-fdef4897-0ef49d91-63d4031d-f508a3e9-12 maps to TransactionID: 1c2d7650-b006e46a-ed071f41-bbbede33-fe78e2dd-38d.]

The CA SiteMinder® Federation system generates a new SAMLTransactionID only if it is acting as the asserting party. These specific activities are:

- When Federation Web Services redirects the browser to the authentication URL to establish a session.
- For the following HTTP-Artifact single sign-on transactions:
  - When the asserting party sends the artifact to the relying party.
  - When the asserting party resolves the artifact.
- When the user is redirected to the Identity Discovery profile URL.
- During single logout at the asserting party.

At the relying party, there exists a request ID, which can be traced easily through the log files. The request ID makes it unnecessary for the CA SiteMinder® Federation system to generate a SAMLTransactionID at the relying party.

For each unique SAML transaction ID, there can be multiple transaction IDs. When a new HTTP transaction occurs, a new transaction ID is generated. This transaction ID is mapped to the single SAML transaction ID. For example, in the trace log you can see the following entries:

```
SamlTransactionID ["xyz"] maps to TransationID["123"]
["123"] HTTP operation
["123"] HTTP operation
```

A new transaction ID "456" is generated:

```
SamlTransactionID["xyz"] Maps to Transactionid["456"]
["456"] <some operation>
["456"] <some operation>
```

Transaction IDs are placed in the fwstrace.log and the smtracedefault.log. The same set of transaction IDs for a single transaction is written to each of these logs. The trail of IDs in these logs enables you to follow a transaction. If there is a failure, the IDs help you determine which event failed for a transaction.

How To Follow a Single Transaction in a Log

To monitor a transaction, you can follow the two types of transaction IDs in the FWSTrace.log or smtracedefault.log. If there is a failure, looking at the IDs can help you determine the failure point.

To follow a transaction in a log, use one or more of the following methods:

- Open the trace file in a text editor and search on the string **SAMLTransactionID** (no spaces) or search for a sepcific SAMLTransactionID. This collection of entries in the log provides a view of the entire end-to-end transaction. You can see how far a transaction proceeded.
- Follow the transaction ID in the log file. The transaction ID represents HTTP transactions. Multiple transaction IDs can be associated with a single SAML Transaction ID. A failed transaction displays the transaction ID in the browser. To search the FWSTrace.log and smtracedefault log for the checkpoint error messages, use the displayed transaction ID.
- Parse the log files with a tool that searches files. On UNIX and Windows platforms, you can use a tool like the grep command. The grep command can stream through raw data, line by line, without your having to load a large text file into a text editor.

  Example:

```
[usr@rhel632 etc]# more fwstrace.log| grep checkpoint
[CHECKPOINT = SSOSAML2_SPCONFFROMPS_REQ]]
[CHECKPOINT = SSOSAML2_SPCONFREAD_REQ]]
[CHECKPOINT = SSOSAML2_SPCONFFROMCACHE_REQ]]
[CHECKPOINT = SSOSAML2_SESSIONCOOKIEVALIDATE_REQ]]
```

Federation Services Trace Logging (smtracedefault.log)

The profiler is the Policy Server facility for logging. You can use the profiler to collect trace messages for federation services and write them to the smtracedefault.log file.

The component that controls the trace messages for federation services at the Policy Server is the Fed_Server component.

The Policy Server Profiler allows you to trace internal Policy Server diagnostics and processing functions.

Follow these steps:

1. Start the Policy Server Management Console.

   Important! If you are accessing this graphical user interface on Windows Server 2008, open the shortcut with Administrator permissions. Use Administrator permissions even if you are logged in to the system as an Administrator. For more information, see the release notes for your CA SiteMinder component.

2. Click the Profiler tab.
3. Set the Enable Profiling option to enable profiling.
4. To select configuration settings for the Profiler, do one of the following:
   - Accept the Profiler settings specified by the default smtracedefault.txt file presented in the Configuration File drop-down list.
   - Select another configuration file that has already been selected during this management session from the Configuration File drop-down list.
   - Click the Browse button to select another configuration file.
5. To change the Profiler settings stored in a Profiler configuration file and save them in the same or a new file, click the Configure Settings button to open the Policy Server Profiler dialog.
6. Adjust the settings presented in the Output group box to specify the output format for information generated by the Policy Server Profiler.
7. Click Apply to save your changes.

Notes:

Changes to the Profiler settings take effect automatically. However, if you restart the Policy Server, a new output file (if the Profiler is configured for file output) is created. The existing Profiler output file is automatically saved with a version number. For example:

```
smtracedefault.log.1
```

If changes to the Logging or Tracing facility settings are not related to the Profiler output file, for example, enabling/disabling the console logging on Windows, the existing file is appended with new output without saving a version of the file.

By default The Policy Server retains up to ten output files (the current file and nine backup files). Older files are replaced automatically with newer files when the ten file limit is reached. You can change the number of files to retain by configuring the TraceFilesToKeep DWORD registry setting to the required decimal value. The TraceFilesToKeep registry setting must be created in the following registry location:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Netegrity\SiteMinder\CurrentVersion\
LogConfig\TraceFilesToKeep
```

The Profiler tab has a "Buffered Tracing" option, which is set by default to improve Policy Server performance. This option is on Solaris systems only.

Federation Web Services Trace Logging (FWSTrace.log)

To simplify the task of collecting tracing data simpler, a series of preconfigured templates are installed with the Web Agent Option Pack. You can use these templates instead of creating your own trace configuration file to collect the data.

The following templates are available:

Template	Tracing Messages Collected
FWSTrace.conf	Default template. Collects data that you specify.
FWS_SSOTrace.conf	Collects single sign-on messages
FWS_SLOTrace.conf	Collects single logout messages
FWS_IPDTrace.conf	Collects Identity Provider Discovery Profile messages

All the FWS templates include the Fed_Client component and subcomponents for the specific data being tracked. To see the exact contents, open each template.

Follow these steps:

1. Navigate to the template directory in *web_agent* or *web_agent_option_pack_home*/config.
2. Make a copy of the template, rename it.
3. (Optional) Modify the template so it includes only the data you want to monitor.

   Note: Do not edit the template directly.
4. Save the new template.

The templates determine the federation components that the federation system monitors. To enable trace logging and format how the data appears in the log file, modify the Logger.Config properties file.

Follow these steps:

1. Navigate to *web_agent or webagent_optionpack_home*/affwebservices/WEB-INF/classes.
2. Open the LoggerConfig.properties file. The LoggerConfig.properties file contains descriptions of all the settings.
3. Set the TracingOn setting to Yes. This option instructs the trace facility to write messages to the log file.

4. Set the TraceFileName setting to the full path of the log file. The default location is in *web_agent or webagent_optionpack_home*/config/FWSTrace.log.

   Note: You can rename the log file. FWSTrace.log is the default name.

5. Set the TraceConfigFile setting to the full path of the trace configuration file. This file can be the default template, one of the other preconfigured templates, or your own configuration file. Regardless of which template you specify, all output is written to the log file you specify in the TraceFileName setting.

   Specify only one template. All the templates reside in the directory *web_agent* or *web_agent_option_pack_home*/config.

6. Optionally, modify how the information in the trace log output file is displayed. The following settings dictate the format of the log file:

   - TraceRollover
   - TraceSize
   - TraceCount
   - TraceFormat
   - TraceDelim

FWS Template Sample

The following text is an excerpt from the FWS_SLOTrace.conf template. Most of the file contains comments and instructions on how to use the file, the command syntax, and the available subcomponents for the Fed_Client component.

The excerpt shows the component, Fed_Client and the subcomponents (Single_Logout and Configuration) that are monitored. The excerpt also shows the specific data fields that indicate the required contents of each message (Date, Time, Pid, Tid, TransactionId, SrcFile, Function, Message).

```
components: Fed_Client/Single_Logout, Fed_Client/Configuration
data: Date, Time, Pid, Tid, TransactionID, SrcFile, Function, Message
```

Chapter 25: Open Format Cookie Details

The federation open format cookie lets applications assert user attributes to CA SiteMinder and consume user attributes that CA SiteMinder encapsulates. The open format cookie has the following general characteristics:

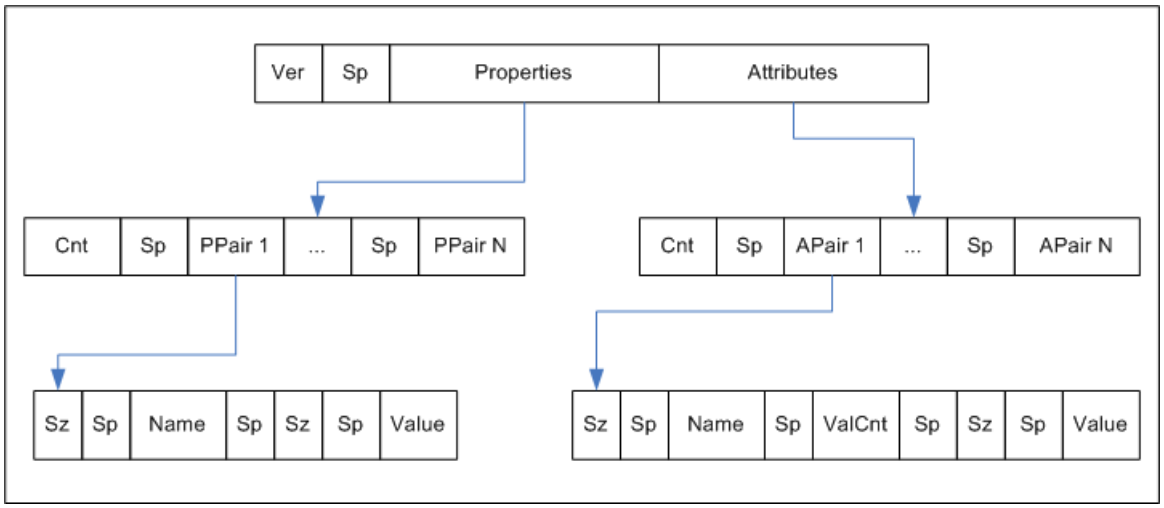
- The cookie is accessible by applications written in any programming language.
- The cookie content consists of a string of UTF-8 bytes, which supports international character sets.
- The combined size in UTF-8 bytes of each name/value pair precedes the name/value pair.
- Space characters are added for legibility.
- The cookie is simple to parse and easily extensible.

Important! If the cookie contains any unsafe characters such as '=', enclose the value in double quotes. You can specify this option through the user interface, or through the SDK.

The open format cookie contains the following property information:

- Cookie Version
- Name ID
- Name ID Format
- Session ID
- AuthnContext
- UserDN (same as User ID)

The following diagram shows the open format:



Key:

- Ver — the cookie format version; for CA SiteMinder® Federation r12.1, this value is 1.
- Sp — an ASCII space character, used only to improve readability.
- Properties — information about the principal.
- Attributes — SAML attributes from the Assertion
- Cnt — the number of name value pairs that follow, represented in ASCII.
- Sz — the length of the name or value that follows
- ValCnt — the number of attribute values that follow. For CA SiteMinder® Federation r12.1, multiple values for an attribute are not supported. Set this value to 1.

The Backus-Naur Form (BNF) for this format is following (0* means 0 or more; 1* means at least 1).

- DIGIT = ASCII digit (0 through 9)
- CHAR = UTF-8 character
- Sp = ASCII space (character 32)
- Token = 1*CHAR
- Cookie = Version Sp Properties Attributes
- Version = 1*DIGIT

- Cnt = 1*DIGIT
- Properties = Cnt 1*PPair
- Attributes = Cnt 0*APair
- ValCnt = 1*DIGIT
- PPair = Sz Sp Name Sp Sz Sp Value
- APair = Sz Sp Name Sp ValCnt Sp Sz Sp Value
- Sz = 1*DIGIT
- Name = Token

Value = Token

Contents of the Open Format Cookie

The federation open format cookie lets applications assert user attributes to CA SiteMinder and consume user attributes that CA SiteMinder encapsulates. The open format cookie has the following general characteristics:

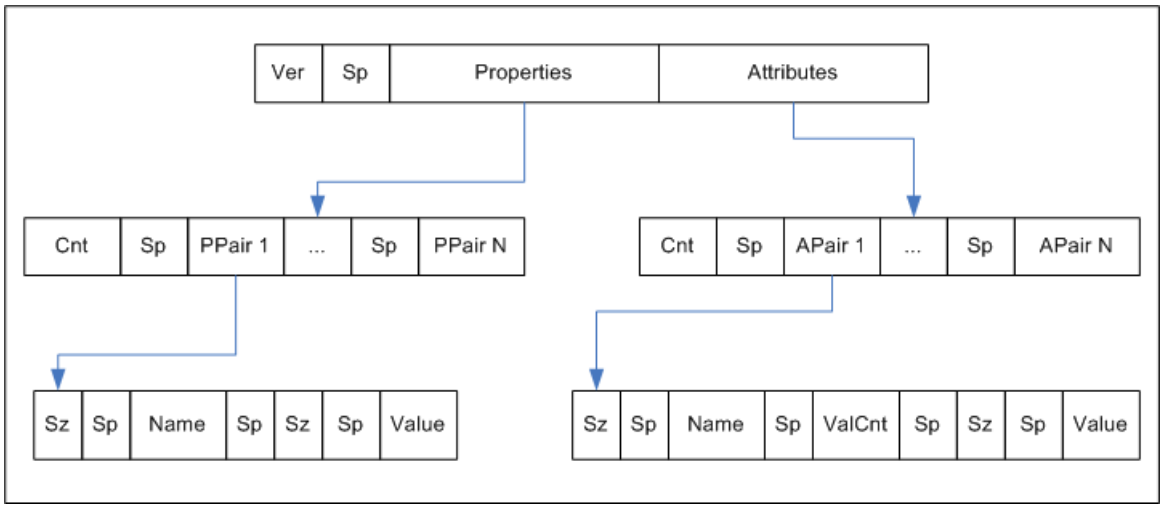
- The cookie is accessible by applications written in any programming language.
- The cookie content consists of a string of UTF-8 bytes, which supports international character sets.
- The combined size in UTF-8 bytes of each name/value pair precedes the name/value pair.
- Space characters are added for legibility.
- The cookie is simple to parse and easily extensible.

Important! If the cookie contains any unsafe characters such as '=', enclose the value in double quotes. You can specify this option through the user interface, or through the SDK.

The open format cookie contains the following property information:

- Cookie Version
- Name ID
- Name ID Format
- Session ID
- AuthnContext
- UserDN (same as User ID)

The following diagram shows the open format:



Key:

- Ver — the cookie format version; for CA SiteMinder® Federation r12.1, this value is 1.
- Sp — an ASCII space character, used only to improve readability.
- Properties — information about the principal.
- Attributes — SAML attributes from the Assertion
- Cnt — the number of name value pairs that follow, represented in ASCII.
- Sz — the length of the name or value that follows
- ValCnt — the number of attribute values that follow. For CA SiteMinder® Federation r12.1, multiple values for an attribute are not supported. Set this value to 1.

The Backus-Naur Form (BNF) for this format is following (0* means 0 or more; 1* means at least 1).

- DIGIT = ASCII digit (0 through 9)
- CHAR = UTF-8 character
- Sp = ASCII space (character 32)
- Token = 1*CHAR
- Cookie = Version Sp Properties Attributes
- Version = 1*DIGIT

- Cnt = 1*DIGIT
- Properties = Cnt 1*PPair
- Attributes = Cnt 0*APair
- ValCnt = 1*DIGIT
- PPair = Sz Sp Name Sp Sz Sp Value
- APair = Sz Sp Name Sp ValCnt Sp Sz Sp Value
- Sz = 1*DIGIT
- Name = Token
- Value = Token

Appendix A: Encryption and Decryption Algorithms

This section contains the following topics:

Open Format Cookie Encryption Algorithms

The open format cookie supports the following options for password-based encryptions:

FIPS_Compat and FIPS_Migration Modes

PBE/SHA1/AES/CBC/PKCS12PBE-1000-128

PBE/SHA1/AES/CBC/PKCS12PBE-1000-192

PBE/SHA1/AES/CBC/PKCS12PBE-1000-256

PBE/SHA256/AES/CBC/PKCS12PBE-1000-128

PBE/SHA256/AES/CBC/PKCS12PBE-1000-192

PBE/SHA256/AES/CBC/PKCS12PBE-1000-256

PBE/SHA1/3DES_EDE/CBC/PKCS12PBE-1000-3

PBE/SHA256/3DES_EDE/CBC/PKCS12PBE-1000-3

FIPS_Only Mode

AES128/CBC/PKCS5Padding

AES192/CBC/PKCS5Padding

AES256/CBC/PKCS5Padding

3DES_EDE/CBC/PKCS5Padding

Digital Signing and Private Key Algorithms

CA SiteMinder uses the following algorithms for partnership signing options.

Encryption Key Algorithms

RSA-V15, RSA-OEAP

Encryption Block Algorithms

3DES, AES-128, AES-256

CA SiteMinder uses the following algorithms for Private Key generation (Certificate/Keys):

Key Algorithm

RSA

Sign Algorithms

MD5withRSA, SHA1withRSA, SHA256withRSA & SHA512withRSA

Back Channel Communication Algorithms

For back channel communication related to HTTP-Artifact single sign-on and SAML 2.0 Single Logout, CA SiteMinder supports the following ciphers, depending upon the FIPS mode:

FIPS_Compat and FIPS_Migration Modes—RC4 and AES

RSA_With_RC4_SHA

RSA_With_RC4_MD5

RSA_With_AES_128_CBC_SHA

RSA_With_AES_256_CBC_SHA

FIPS_Only Mode—AES only

RSA_With_AES_128_CBC_SHA

RSA_With_AES_256_CBC_SHA

Java SDK Encryption Algorithms

The CA SiteMinder® Federation Java SDK supports the following encryption algorithms:

Without a Password

"AES/CBC/PKCS5Padding"

With a Password

"PBE/SHA1/AES/CBC/PKCS12PBE-5-128"

Crypto Algorithm

FMCrypto Encryption/Decryption Algorithm

AES_128