CA SiteMinder®

Federation Manager Guide: Partnership Federation

r12.5

CA Technologies Product References

This document references the following CA Technologies products:

- SiteMinder®

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.

Contents

Chapter 1: Partnership Federation Introduction

This section contains the following topics:

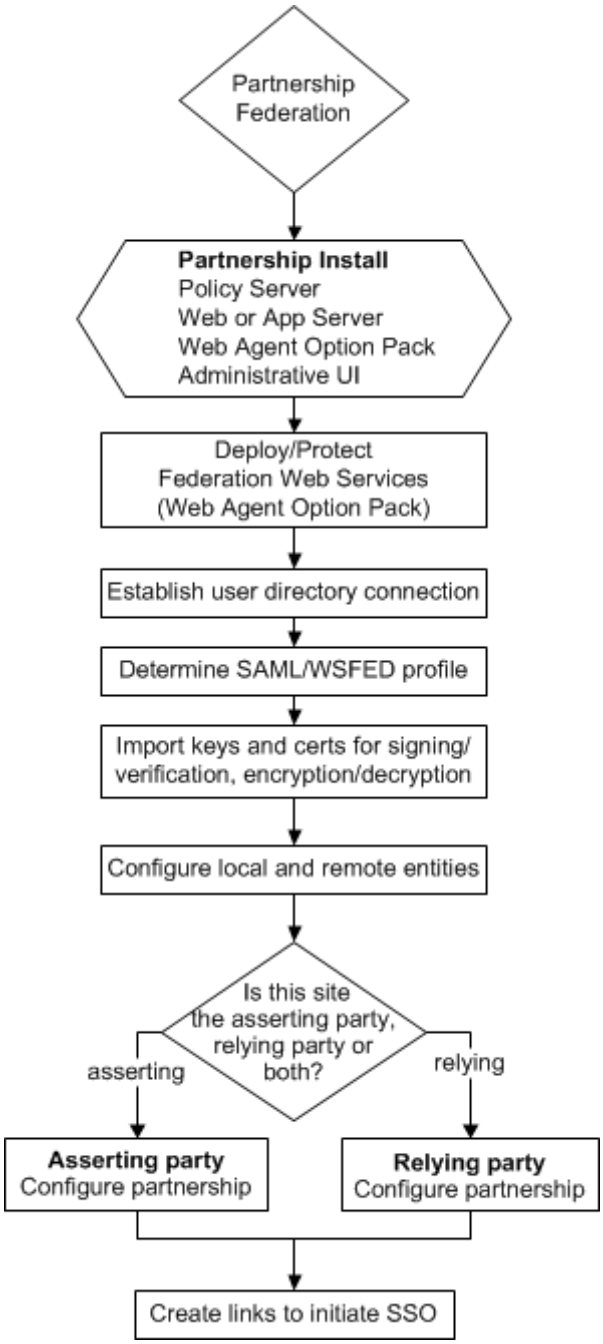
Overview

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.

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 SAML 1.1 and SAML 2.0 protocols.

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, 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 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 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 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 SiteMinder provisioning methods. Remote provisioning uses an independent provisioning application to establish a user record. To pass assertion data, 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 fundamentals
- SAML bindings POST and artifact
- SAML profiles, such as Single Sign-on (SSO), Single logout (SLO), and Enhanced Client or Proxy (ECP)
- Public Key Infrastructure (PKI) fundamentals
- Secure Socket Layer communication basics

Terminology Used in this Guide

The following terms are used in this guide:

Asserting Party

A SAML authority that generates an assertion for use by a relying party. The asserting party creates, maintains, and manages identity information for users and provides user authentication to other relying parties. For SAML 1.1, the asserting party is known as the Producer. For SAML 2.0, the asserting party is known as the Identity Provider.

Important! In this guide, the term *asserting party* is used to mean a producer or an Identity Provider.

Assertion Consumer Service (SAML 1.1 and 2.0)

A Service Provider component that receives a SAML artifact or an HTTP form with an embedded SAML response and obtains the corresponding SAML assertion. The Assertion Consumer Service issues partnership federation session cookies, and if you are integrating with SiteMinder, a SiteMinder session cookie.

Assertion Retrieval Service (SAML 1.1)

A Producer-side service that handles SAML 1.1 authentication using HTTP Artifact binding. This service retrieves the assertion at the Producer.

Artifact Resolution Service (SAML 2.0)

An Identity Provider-side service that performs SAML 2.0 authentication using the HTTP Artifact binding. This service retrieves the assertion at the Identity Provider.

AuthnRequest Service (SAML 2.0)

A service that enables a Service Provider to generate an AuthnRequest message for cross-domain single sign-on. This message contains information that enables the Service Provider to send the browser to the Single Sign-on Service at the Identity Provider. The AuthnRequest service is used for single sign-on using POST and artifact binding.

Note: The format of the AuthnRequest message that the service issues is specified in the Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0.

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 Federation Manager 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 Federation Manager Java SDK can encrypt the cookie using a FIPS-compliant (AES) algorithm or a non-FIPS (PBE) algorithm. The Federation Manager .NET SDK can encrypt the cookie using only a FIPS-compatible algorithm.

Relying Party

A SAML entity that uses information from a SAML authority to provide access to services. The relying party uses assertions from an asserting party to authenticate a user. For SAML 1.1, the relying party is known as the Consumer. For SAML 2.0, the relying party is known as the Service Provider.

Important! In this guide, the term *relying party* is used to mean a consumer or a Service Provider.

Single Logout Service (SAML 2.0)

This service allows a user to log out from all applications in the federation simultaneously with a single logout event. An Identity Provider or a Service Provider can initiate single logout.

Single Sign-on Service (SAML 1.1 and SAML 2.0)

For SAML 1.1, the SSO service enables a Producer to process Producer-initiated requests for federated resources.

For SAML 2.0, the SSO service enables an Identity Provider to process IdP-or SP-initiated requests for federated resources.

The Producer/IdP gathers the necessary information from the Consumer/SP to generate an assertion, which it passes back to the Consumer/SP. The Consumer/SP then uses the assertion for authentication.

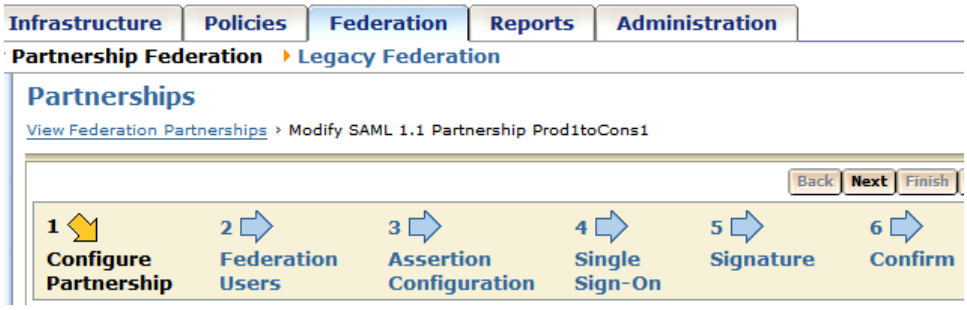
Unified Expression Language

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.

The following figure shows an example of a partnership configuration wizard:



Chapter 2: Prerequisites for Partnership Federation

This section contains the following topics:

Prerequisites for a SiteMinder Asserting Partner

For 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 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 SiteMinder Relying Partner

For 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 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 SiteMinder is installed at both sites in the partnership. However, you can have SiteMinder at one site and a different SAML-compliant product at the other site and still engage in a partnership.

With 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 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 SiteMinder is at both sites:

1. Establish a user directory connection.
2. Create the local and remote entities.
3. Configure the local IdP-to-SP partnership definition at the IdP.
4. Configure the local SP-to-IdP partnership definition at the SP.
5. Activate the partnership.
6. 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

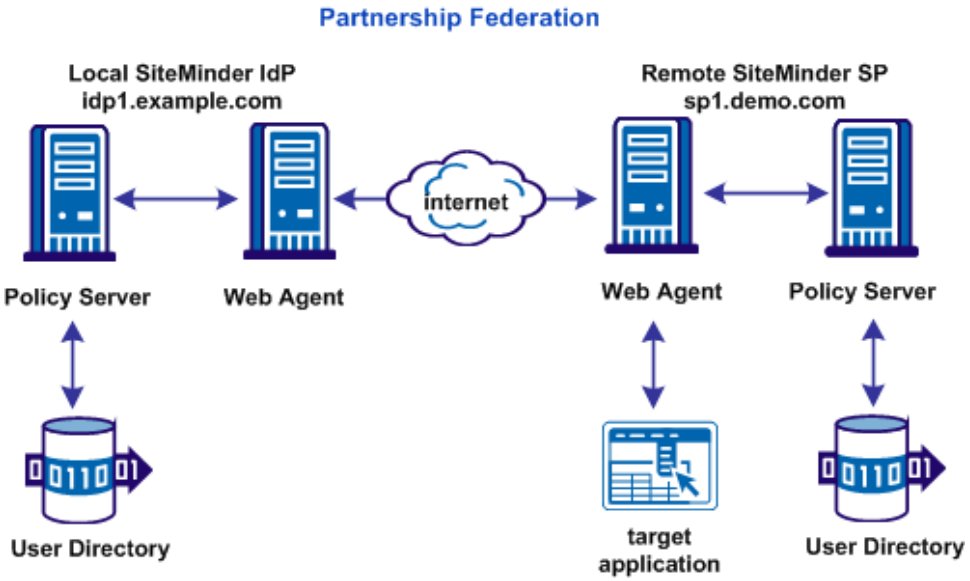
http://idp1.example.com:9090/affwebservices/public/saml2sso

Assertion Consumer Service URL at the SP

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

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

The following figure shows the sample partnership with 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 SiteMinder 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.

Deploy the Federation Web Services Application on a Server

The Web Agent Option pack installs the Federation Web Services (FWS) application. FWS is a required component for SiteMinder legacy and partnership federation.

Configure the FWS application for the sample deployment. This sample deployment uses WebLogic as the application server where you deploy FWS.

To set up FWS

- Install the JDK for Federation Web Services.
- Deploy the FWS application on an WebLogic Application Server.
- Configure the AffWebServices.properties File at the IdP.

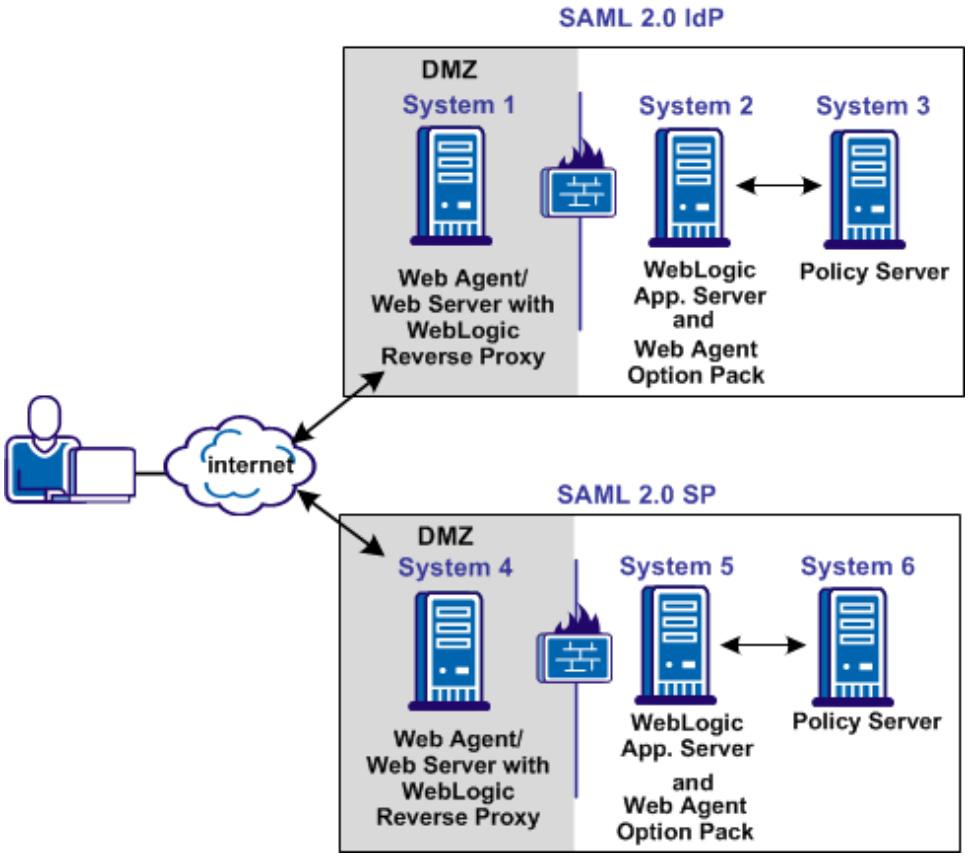
Install the JDK for Federation Web Services

The Web Agent Option Pack requires a JDK to run the Federation Web Services application.

For the correct JDK version, go to the Technical Support site and search for the SiteMinder Platform Support Matrix for the release.

Deploy FWS on a WebLogic Application Server

The following illustration shows a SiteMinder and WebLogic sample configuration. The illustration provides an example of how to deploy FWS in a sample federated environment.



In this environment, deploy the FWS application is on System 2 and System 5.

Important! Complete the deployment procedure for the Web Agent at the IdP and the SP.

After installing the software components on the systems in the illustration, deploy the FWS application. Deploy the application on System 2 for the asserting party and on System 5 for the relying party.

To deploy the FWS application

1. Set the LD_LIBRARY_PATH variable
2. Create a SmHost.conf file
3. Create a WebAgent.conf file
4. Modify the AffWebServices.properties file

5. Configure the WebLogic Reverse Proxy Plug-in
6. Deploy the FWS Application on WebLogic

Important! For the FWS application to work with WebLogic Server 8.1.4 or higher, review the weblogic.xml file in the WEB-INF directory. Verify that the prefer-web-inf-classes parameter is set to true. The weblogic.xml file is located in the following directory:

webagent\affwebservices\WEB-INF

The following code excerpt shows how to set the prefer-web-inf-classes parameter:

```
  <weblogic-web-app>
  <container-descriptor>
    <prefer-web-inf-classes>true</prefer-web-inf-classes>
  </container-descriptor>
</weblogic-web-app>
```

If prefer-web-inf-classes is set to false, change the value to true.

Source the Environment Script on a UNIX Operating Environments

After you install the Web Agent Option Pack on a UNIX system, the installation program creates an environment script (ca-wa-opack-env.sh).

Source the environment script so the library path of the application server points to the location of the Web Agent Option Pack /bin directory.

Source the script by entering the following command at the command line:

```
. ./ca-wa-opack-env.sh
```

Setting the correct library path lets the option pack and the web or application server to work together.

After you source the script, the library path is set. The variable name for the library path differs depending on the operating system. Example of several library paths:

Solaris/Linux

LD_LIBRARY_PATH=/*webagent_option_pack_home*/bin

HP-UX

SHLIB_PATH=/*webagent_option_pack_home*/bin

AIX

LIBPATH=/*webagent_option_pack_home*/bin

Important! The application server startup script can reset the library path. Ensure that the path to the Web Agent Option Pack is the first entry in the path.

The path to the Web Agent Option Pack environment script points to one of the following locations:

- The installation directory of the web agent option pack. The default location is: */webagent_option_pack_home*/bin.
- The installation directory of the web agent.

  If you install the option pack on the same system as the web agent, the script resides in the web agent directory. For any UNIX installation, the default location is */web_agent_home*/bin.

Create an SmHost.conf File

The FWS application requires an SmHost.conf file. However, the Web Agent Option Pack does not install this file, so you must create it.

To create an SmHost.conf

1. Go to the directory */webagent_option_pack_home*/bin
2. Run the smreghost.exe.

   For instructions on running smreghost.exe, see the *Web Agent Installation Guide*.
3. Put the SmHost.conf file in the following directory on Systems 2 and 5:

   /webagent_option_pack_home/config

Create a WebAgent.conf File

The FWS application requires the WebAgent.conf file. However, the Web Agent Option Pack does not install this file, so you must create it.

To create a WebAgent.conf file

1. Copy the WebAgent.conf file from System 1 to the following directory on System 2 and System 5:

   /webagent_option_pack_home/config

   webagent_option_pack_home

   Defines the installed location of the Web Agent Option Pack on System 2 or System 5.
2. Modify the WebAgent.conf file by:
   a. Setting the EnableWebAgent parameter to YES.
   b. Modifying other configuration parameters to suit FWS.

The following sample shows a WebAgent.conf file for the FWS application:

```
# WebAgent.conf - configuration file for the Federation Web Services Application
#agentname="agent_name, IP_address"
HostConfigFile="/webagent_option_pack/config/SmHost.conf"
AgentConfigObject="agent_config_object_name"
EnableWebAgent="YES"
```

Note: The Agent Configuration Object referenced in this WebAgent.conf file must be a new object that you create. Do not specify the object in use by the Web Agent installed in your environment.

Specify the Location of the WebAgent.conf File

The AffWebServices.properties file contains all the initialization parameters for Federation Web Services. For deploying FWS, set only the parameter that specifies the location of the WebAgent.conf file.

Follow these steps:

1. Navigate to the AffWebServices.properties file. Locate this file in the following directory:

   web_agent_optionpack_home/affwebservices/WEB-INF/classes

2. Set the AgentConfigLocation parameter to the location of the WebAgent.conf file at each partner site.

   - Windows example:

     C:\\Program Files\\CA\\webagent_optionpack\\config\\WebAgent.conf

     Note: Federation Web Services is a Java component, so the Windows paths must contain double backslashes.

   - UNIX example:

     *web_agent_optionpack_hom*e/config/WebAgent.conf

3. Repeat this procedure for each application server where the Web Agent Option Pack is installed.

4. Accept the default values for the rest of the settings in the properties file.

Deploy the FWS Application on WebLogic

Deploy the FWS application on System 2 and System 5.

Important! For the FWS application to work with WebLogic Server, review the weblogic.xml file in the WEB-INF directory. Verify that the prefer-web-inf-classes parameter is set to true.

The weblogic.xml file is located in the directory webagent\affwebservices\WEB-INF.

The following code excerpt shows how to set the prefer-web-inf-classes parameter:

```
<weblogic-web-app>
 <container-descriptor>
  <prefer-web-inf-classes>true</prefer-web-inf-classes>
 </container-descriptor>
</weblogic-web-app>
```

In addition, verify that the precompile parameter is set to true, as listed following:

```
<jsp-descriptor>
   <precompile>true</precompile>
<jsp-descriptor>
```

Follow these steps: to deploy FWS

1. Use the WebLogic Server Console and deploy FWS. The FWS application is installed in:

   /webagent_option_pack_home/affwebservices/

   For more information about deploying a web application, see the WebLogic documentation.

2. Test that the FWS application is working. Open a web browser and enter:

   http://*fqhn*:*port_number*/affwebservices/assertionretriever

   fqhn

   Defines the fully qualified host name.

   port_number

   Defines the port number of the server where the Federation Web Services application is installed.

For example:

http://myhost.ca.com:81/affwebservices/assertionretriever

If Federation Web Services is operating correctly, you see the following message:

```
Assertion Retrieval Service has been successfully initialized.
The requested servlet accepts only HTTP POST requests.
```

This message indicates that Federation Web Services is listening for data activity. The FWS application is now deployed for the WebLogic server.

If Federation Web Services is not operating correctly, a message that the Assertion Retrieval Service has failed displays. If the service fails, review the Federation Web Services log.

Note: For instructions on enabling trace logging for the FWS application, see Trace Logging.

Configure the WebLogic Reverse Proxy Plug-in

To set up the WebLogic Reverse Proxy plug-in:

1. On System 1, configure the WebLogic reverse proxy plug-in on the Apache Web Server.

   For more information, see WebLogic documentation.

2. Add the following aliases to the configuration file of the web server.

   This example uses the Apache httpd.conf file.

```
<IfModule mod_weblogic.c>
WebLogicHost <WebLogic_Machine_IP Address>
WebLogicPort <WebLogic_Machine_Port_Number>
</IfModule>

<Location /affwebservices>
SetHandler weblogic-handler
Debug ALL
</Location>
```

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.

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 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/siteminderagent/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 a SiteMinder 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. 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 SiteMinder to replace the variable (%s) with the value of the Name ID attribute from the assertion. 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 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.

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, 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 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 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 57)
Partnership Creation and Activation (see page 73)

Chapter 4: 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 5: 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.
- Create an entity by importing metadata.

How to 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. Select the asserting or relying party as the New Entity Type.
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.

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

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.

   Note: Click Help for a description of fields, controls, and their respective requirements.
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 76) for more details about entities within a partnership.

How to Create an Entity by Importing Metadata

You can import data from a metadata file to create a federation entity. Importing SAML 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 to SiteMinder from another federation product.

  Note: SiteMinder 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 to serve as the basis 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 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.

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

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: SiteMinder 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.

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

Exporting a Local Entity

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 metadata file can be imported to create a partnership or remote entity.

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 adheres to the SAML specification. When you import the SAML 1.1 data, the terms are imported correctly using SAML 1.1 terminology.

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.

   Note: When you export metadata from a local entity, you are asked to specify a new partnership name.
4. Complete the 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, controls, and their respective requirements.
5. Click Export.

6. A dialog prompting you to open or save the metadata file displays.

   Only open it to view it.

7. 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.

Chapter 6: Key and Certificate Management for Federation

In a federation environment, SiteMinder uses a key/certificate pair and standalone certificates for a number of functions:

- Signing/verification of assertions
- Signing/verification of authentication requests (SAML 2.0 only)
- Signing/verification of single logout requests and responses (SAML 2.0 only)
- 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

The *Policy Server Configuration Guide* contains overview information and instructions about managing keys and certificates for SiteMinder.

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 you have installed the SiteMinder Web Agent.

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://.

Chapter 7: Storing User Session, Assertion, and Expiry Data

This section contains the following topics:

Federation Data Stored in the Session Store

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

- HTTP-POST single use policy (SAML 2.0)

  The single use policy feature prevents assertions (POST binding) 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 helps ensure 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.

- Authentication Session Variables Persistence (SAML 1.x and SAML 2.0)

  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 (SAML 1.x and SAML 2.0)

  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 attributes that are extracted from an assertion in the session store so they can be supplied as HTTP header variables.

Enable the Session Store

Enable the session store to store 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.

The session server database is where the Policy Server Session Server stores persistent session data.

Follow these steps: to enable a session store

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 SiteMinder 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.

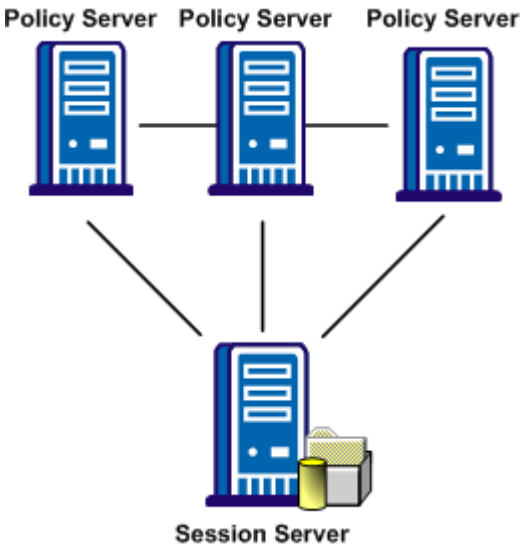
- For SAML 2.0 single logout, share the session store at the asserting and relying party across all Policy Servers in the cluster.

  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 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 8: Partnership Creation and Activation

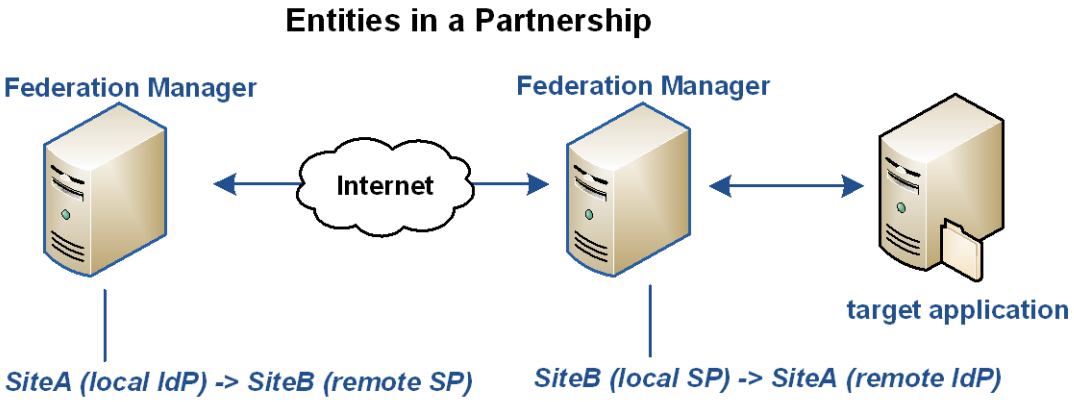
This section contains the following topics:

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 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 SiteA, SiteA is a local Identity Provider (IdP) and SiteB is the remote Service Provider (SP). For the partnership configuration at SiteB, SiteB is the local Service Provider (SP) and SiteA 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 role is local, and which federation role is remote.

To specify the partnership type

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.

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

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.

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.

3. Specify a user name or create a filter for the User Name/Filter By column.

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 an additional user class for the same directory or another user directory.

The selection of users is complete.

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 78), which lets an asserting party create a user identifier.

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.

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:

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

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
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.

Assertion Configuration

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

Name ID

The Name ID attribute, which is required in an assertion, identifies a user in a unique way. The format of the Name ID establishes the type of content that the assertion uses as the source of the ID.

Assertion Attributes

An attribute statement passes user attributes, DN attributes, or static data in an assertion to the relying party. When the relying party receives the assertion, it makes the attribute values available to applications.

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 included in the assertion in an <AttributeStatement> element or an <EncryptedAttribute> element. Attributes take the form of name/value pairs and can be made available as HTTP Headers or HTTP Cookies.

Note: Attributes statements are not required in an assertion.

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 SiteMinder.

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

Configure Assertion Options

Configure assertion options in the Assertion Configuration step of the partnership wizard.

Follow these steps:

1. Configure the settings in the Name ID section.

   The relying party uses these values to know how to interpret the value that is passed in the assertion.

   Based on the value of the NameID Type, complete one of the following tasks:

   - If you selected Static or User Attribute for the Name ID type, complete the Value field.
   - If you selected the DN Attribute for the Name ID type, complete the Value and the DN specification fields.

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

2. (Optional - SAML 2.0 only) Select Allow Creation of User Identifier so the asserting party can create a value for the NameID. 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.

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

   Click Help for detailed information about the columns in the attribute table.

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

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

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

5. Click Next to continue with partnership configuration.

User Consent at a SAML 2.0 IdP

A 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, 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

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.

Single Sign-on Configuration (Asserting Party)

Configure single sign-on at the asserting party to specify how the asserting party delivers an assertion to a relying party.

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

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

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

2. Complete the Authentication Class field. You can supply a static URI for SAML 1.1 and SAML 2.0. Additionally, for SAML 2.0 only, SiteMinder can automatically detect an authentication class. The URI is placed in the AuthnContextClassRef element in the assertion to describe how a user is authenticated.

3. Complete the fields in the SSO section to determine how single sign-on operates. 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. Read the information about assertion validity (see page 89) to understand how these settings work together.

   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

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

4. Specify the URL for the Remote Assertion Consumer Service. This service is the service at the relying party that processes received assertions.

5. If you selected HTTP-Artifact, configure the back channel settings (see page 91).

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

   - Enable IDP Discovery Profile (see page 101).
   - Specify status redirect URLs (see page 88) for specific HTTP errors.

More information:

Legacy Artifact Protection Type for the HTTP-Artifact Back Channel (see page 85)
Enhanced Client or Proxy Profile (ECP) (see page 99)
Status Redirects for HTTP Errors (SAML 2.0 IdP) (see page 88)
Single Sign-on Initiation (SAML 2.0) (see page 88)

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 related aspects of how the relying party handles communication.

At the relying party, SiteMinder uses the skew time for the partnership to determine whether the assertion it receives is valid. Read more about assertion validity (see page 89) to understand how SiteMinder uses the configured skew time.

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

2. Configure the settings in the SSO section of the dialog. These settings let you control the following features:

   Single sign-on binding

   For SAML 2.0, you can select HTTP-Artifact and HTTP-POST. If the SP 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 IdP initiates single sign-on, the IdP indicates the binding in use for that particular transaction.

   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
   - User consent requirement

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

3. If you select HTTP-Artifact, configure the authentication method for the back channel in the Back Channel section of the dialog.

   Note: For SAML 2.0, configure the outgoing back channel.

Configuration for single sign-on is complete.

More information:

Enhanced Client or Proxy Profile (ECP) (see page 99)

Status Redirects for HTTP Errors (SAML 2.0 IdP)

For the Identity Provider, you can configure how 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 SiteMinder redirects the user.
4. For each URL, select the redirect method, 302 No Data or HTTP Post.

Redirect handling is configured.

Single Sign-on Initiation (SAML 2.0)

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 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 SiteMinder is acting as an SP, it ignores the SessionNotOnOrAfter value. Instead, a 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.

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

Configure the HTTP-Artifact Back Channel

Protect the HTTP-artifact back channel across which the asserting party sends the assertion to the relying party.

Consider the following limitation:

You cannot use client certificate authentication with the following web servers running ServletExec:

- IIS web servers at a SiteMinder producer/Identity Provider because of a limitation in IIS.
- SunOne/Sun Java Server web servers at a SiteMinder producer/Identity Provider because of a documented limitation in ServletExec.

Follow these steps:

1. Begin at the Back Channel section in the Single Sign-on or the SSO and SLO step of the partnership wizard.
2. Select HTTP-Artifact in the SSO section.

   The Authentication Method field becomes active.
3. Select the type of authentication method for the incoming or outgoing back channel, or both.

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

   - If you select the client certificate authentication scheme, add a private key/certificate pair to the certificate data store. The private key/certificate pair is issued from a Certificate Authority.

     Important! The CN of the Subject in the certificate must be the same as the partnership name in the producer to consumer partnership that is configured at the producer.

     For instructions on adding a certificate, see the Policy Server Configuration Guide. Skip this step if the key/certificate pair is already in the data store.
   - 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. You can enable SSL on each side of the connection for added security.

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 SiteMinder is at the IdP

When SiteMinder is at the IdP, configure the partnership to include an HTTP Redirect-based SLO Service URL and a SOAP-based SLO Service URL.

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 SiteMinder is at the SP

If 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 SiteMinder, that link is:

http://s*p_host:port*/affwebservices/public/saml2slo

This embedded link causes 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, SiteMinder takes its system time. The resulting time becomes the IssueInstant set in the request message. To determine when the logout request expires, 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 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 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 94) where SOAP and HTTP-Redirect are supported.

Follow these steps: to configure SLO

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 97).

3. Configure any of the additional SLO settings:
   - SLO Confirm URL
   - SLO Validity Duration (Seconds)
   - Relay State overrides SLO Confirm URL

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

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 SiteMinder to generate a SAML <LogoutRequest> message. In addition, the SLO Service URL tells SiteMinder 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 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.

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

   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.

Local Logout at the SP (SAML 2.0)

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.

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?Loc
alLogout=true
```

Enhanced Client or Proxy Profile (ECP)

The Enhanced Client or Proxy Profile (ECP) is an application of the SAML 2.0 single sign-on profile. An enhanced client can be 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.

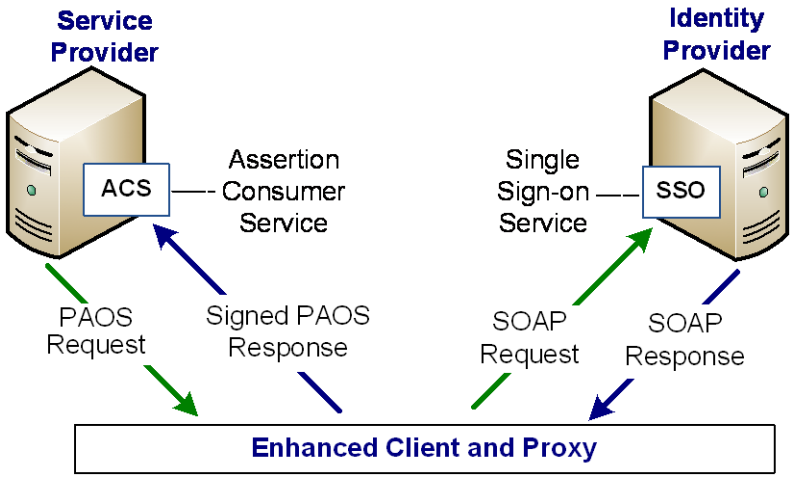
An enhanced client or proxy is a system entity that knows how to contact an Identity Provider and supports the Reverse SOAP binding, PAOS. The ECP acts as the intermediary between the Service Provider and the Identity Provider.

The ECP profile allows the Service Provider to make an authentication request without knowing the Identity Provider. PAOS lets the relying party obtain the assertion through the ECP, which is always directly accessible.

You can enable the ECP profile with single sign-on in the following situations:

- For a Service Provider that expects to service enhanced clients or proxies that require this profile.
- When the Identity Provider and Service Provider cannot communicate directly.
- When a proxy server is in use, such as a wireless access protocol (WAP) gateway in front of a mobile device with limited functionality.

The flow of the ECP profile is shown in the following figure:



To enable the ECP profile

1. Verify that ECP request is directed to the AuthnRequest service at the Service Provider. The following URL shows an example:

   https://*host:port*/affwebservices/public/saml2authnrequest

2. Verify that the headers in the ECP request include attributes that the SAML 2.0 specification requires. The following attributes are examples:

   Accept: text/html; application/vnd.paos+xml

   PAOS: ver='urn:liberty:paos:2003-08';

   'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'

3. Use the user interface to configure single sign-on.
4. Select the Enable Enhanced Client and Proxy Profile check box as part of the single sign-on configuration.

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 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.

   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.

Sign and Encrypt Federation Messages

Securing an assertion and encrypting data within the assertion is a critical part of partnership configuration. The Signature step (SAML 1.1) and the Signature and Encryption step (SAML 2.0) let you configure signing and encryption of assertions.

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

The Signature step lets you define how SiteMinder uses private keys and certificates to verify SAML assertions and assertion responses.

Note: SAML 1.1 does 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. Begin by selecting the Signature step in the partnership wizard.

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

2. 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.

3. For the Artifact and Post signature options, select the specific components (assertion, response) that you want signed.

Note: If you are using SiteMinder in a test environment, you can disable signature processing to simplify testing. Click the Disable Signature Processing checkbox.

Signature configuration at the SAML 1.1 producer is complete.

Signature Configuration at the SAML 1.1 Consumer

The Signature step lets you define how SiteMinder uses private keys and certificates to verify SAML assertions and assertion responses.

Note: SAML 1.1 does 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. Begin by selecting the Signature step in the partnership wizard.
2. 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 checkbox.

Signature configuration at the SAML 1.1 consumer 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.

  Note: 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 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 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 checkbox.

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 SiteMinder uses private keys and certificates to do the following tasks:

- Sign and verify SAML assertions, assertion responses, and authentication requests.

  Note: 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.

   The W3C XML Syntax and Processing standards define these algorithms.

   Important! 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 SiteMinder 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.

   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 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.

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, configure the additional required settings and optional settings.

     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 Federation Manager 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 Federation Manager 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, 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 111) and how to protect the headers.

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

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 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 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 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)

SiteMinder can pass assertion data using HTTP headers.

Follow these steps:

1. Verify that the 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 SiteMinder adds to HTTP headers. Setting a prefix protects HTTP headers against manipulation by an unauthorized user before the SiteMinder consumes an assertion. As a result, only legitimate headers get passed to the target application. Read more about protecting HTTP headers (see page 112).

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.

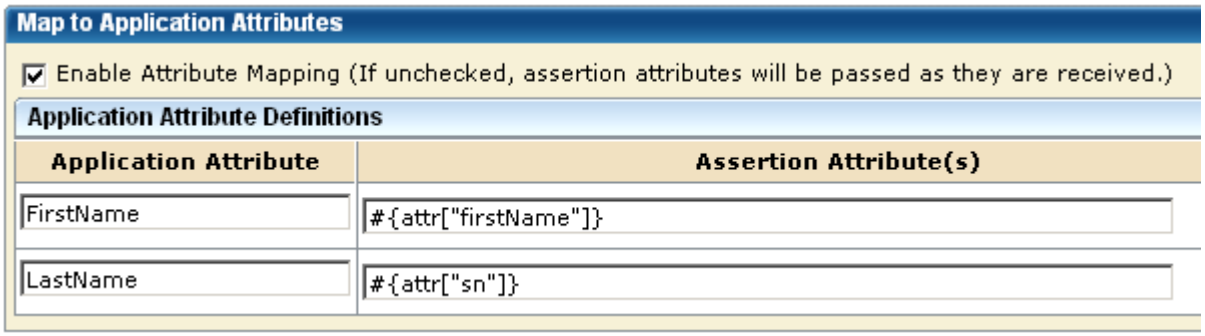
- Transform separate attributes and their values into a single attribute.

  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.

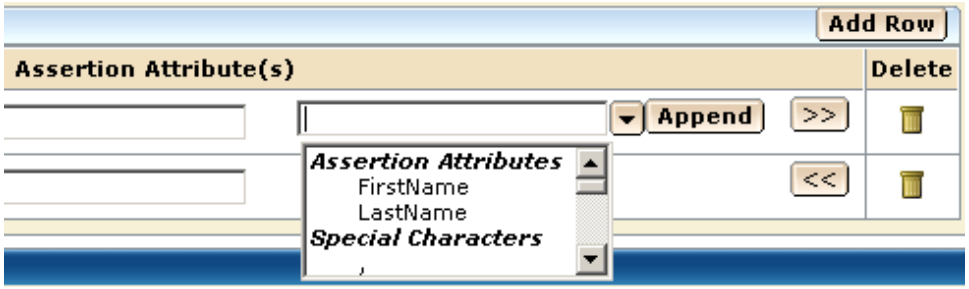
Gather the names of the application and assertion attributes from the necessary parties before setting up attribute mapping.

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 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, SiteMinder places the data in a legacy cookie, an open format cookie, or an HTTP header. 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, 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.

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 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 SiteMinder can apply to the assertion attributes. 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.

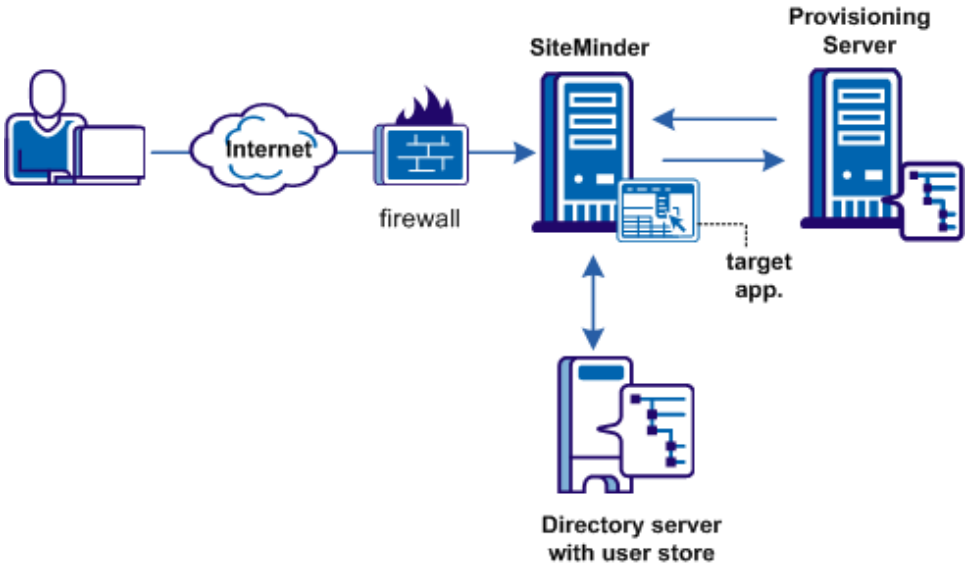
User Provisioning at the Relying Party

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 new user account and then pass the necessary information back to SiteMinder. SiteMinder 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 logging in, the application reads the cookie or the headers and uses the assertion data and the login credentials to establish a user account.

   The provisioning application can consume the open-format cookie using the SiteMinder Federation Java or .NET SDK
5. The user is sent back to the Policy Server and 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 ? federation service

6. The Policy Server attempts user disambiguation a second time. Assuming provisioning is successful, the user is authenticated and cookies or headers are sent to the target application.

   The data delivery method to the target application is defined by the redirect mode you select for the target application.

7. The user is redirected to the target resource.

Delivery of Assertion Data to the Provisioning Application

To accomplish remote provisioning, SiteMinder redirects the browser with the assertion data to the provisioning application.

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 SiteMinder system and the remote provisioning system must be in the same domain.

The cookie can be created in one of two ways:

- A Federation Manager SDK creates the cookie.

  If you select one of the FIPS algorithms (AES algorithms), use a Federation Manager 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 Federation Manager 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 Federation Manager 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.

HTTP Headers

SiteMinder can also pass assertion information as HTTP headers. If you use HTTP headers, the SiteMinder system and the remote provisioning system can be in different domains.

Learn more about using HTTP headers to pass assertion data (see page 111) 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, 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 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.

   Note: Click Help for a description of fields, controls, and their respective requirements.
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 SiteMinder to redirect the user to different applications (URLs) for further processing. For example, when user disambiguation fails, you can configure SiteMinder to send the user to a provisioning system. Setting up redirect URLs is optional and is only configurable at the relying party.

To configure the redirect URLs

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.

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

2. For each redirect option you configure, specify the method by which 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.

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. 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.

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

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.

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.

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.

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 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 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 SiteMinder.

SP_ID

Specifies the Entity ID of the Service Provider defined in the partnership.

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.

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.

Note: SiteMinder Identity Providers do not support the IsPassive query parameter. A third-party Service Provider can include the IsPassive parameter in an AuthnRequest message.

When a Service Provider includes ForceAuthn or IsPassive in the AuthnRequest, a SiteMinder Identity Provider handles these query parameters as follows:

ForceAuthn Handling

When a Service Provider includes ForceAuthn=True in the AuthnRequest message, a SiteMinder Identity Provider challenges the user for their credentials. The challenge happens even when a SiteMinder session does not exist. If the user successfully authenticates, a session is established.

IsPassive Handling

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 SiteMinder session, the Identity Provider returns an error message. SiteMinder requires a session.
- If IsPassive=True in the AuthnRequest message and there is a SiteMinder session, the Identity Provider returns the assertion.
- If IsPassive and ForceAuthn are in the AuthnRequest message and both are set to True, the 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 Federation Manager.

IdP_ID

Specifies the identity that is assigned to the Identity Provider.

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 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.

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 Federation Manager, 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

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, 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, SiteMinder passes a request for an AuthnRequest message.

Chapter 9: Authentication Context Processing

This section contains the following topics:

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 an 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

A SiteMinder Service Provider requests the authentication context by including the <RequestedAuthnContext> element in the authentication request to the Identity Provider. Inclusion of this element is based on a configuration setting in the SP->Identity Provider partnership.

Obtaining the Authentication Context

A SiteMinder Identity Provider obtains the authentication context for a user in one of two ways:

- You specify a static AuthnContext URI in the IdP->SP partnership configuration.
- If you are communicating with a SiteMinder Service Provider that does not support AuthnContext requests, manually enter a URI.
- The AuthnContext URI is determined dynamically using a configured authentication context template.

  SiteMinder maps the authentication context URIs to SiteMinder authentication levels. SiteMinder 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 sent in the request from the Service Provider. If the comparison is successful, the Identity Provider includes the authentication contexts in the assertion it returns to the Service Provider. If validation is configured, 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—only available if the SiteMinder Connector is enabled for the IdP-to-SP partnership.

     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

Configure an Authentication Context Template

Configuring how authentication context information is processed involves the following tasks:

- Set up authentication context templates.
- Obtain AuthnContext information at the IdP.
- Enable the authentication context request at the SP.

An authentication context template defines the specific SAML 2.0 AuthnContext URIs that a partnership supports. Each URI identifies the context class that provides additional information in an assertion.

The template maps the URIs to the protection levels associated with a user session. The protection levels indicate the strength of the authentication, from 1 through 1000, with 1000 being the strongest. An administrator assigns protection levels as part of the authentication scheme that authenticates a user and establishes a user session.

You can select a template on a per-partnership basis; multiple partnerships can use a single template.

Set Up an Authentication Template

Set up an authentication context template to associate URIs and protection levels. This procedure is the same regardless of whether SiteMinder is acting as an Identity Provider or Service Provider.

Follow these steps:

1. Log in to the Administrative UI.
2. Navigate to Federation, Partnership Federation, Authentication Context Template.
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. Click Next.

7. Map the protection levels from an authentication scheme to URIs. SiteMinder protection levels indicate the strength of an authentication, ranging between 1 through 1000, with 1000 being the strongest.

   Note the following information:

   - 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 (see page 139).

8. (Optional) Group URIs that require the same protection level. Use the Change Grouping arrow to move a URI into or out of a group.

   Group URIs by indenting one URI under the previous URI. Individual URIs can have unique protection levels; however, grouping URIs means that they share the level of strength.

9. Click Next to move to the last step of the wizard.
10. Select Finish to confirm the configuration.

The template is complete.

Protection Level Assignments for a Context Template

In the authentication context template, assign a maximum protection level to each URI. The minimum protection level is automatically calculated based on the maximum level for the subsequent URI in the list.

Protection Level Example

Each protection level is mapped to a URI strength level. The original list of URIs can be organized like the following example:

URI	Protection Level Max	URI Strength
urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession	1000	5
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocoPassword	800	4
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol	700	3
urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos	500	2
urn:oasis:names:tc:SAML:2.0:ac:classes:Password	200	1

If you group several of the URIs from the previous table, the grouping enables URIs with different protection levels to have the same URI strength. The modified table that follows shows the groupings.

URI	Protection Level Max	URI Strength
urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession	1000	3
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocoPassword	800	3
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol	700	2
urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos	500	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 is 3.

Configure Authentication Context Processing at the IdP

The SiteMinder 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.

  SiteMinder 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 136).

Specify How the IdP Obtains the Authentication Context

Create an authentication context template to obtain the authentication class automatically.

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 that is based on the session context and the 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.

Configure AuthnContext	Ignore RequestedAuthnContext	SP requests AuthnContext	Result
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.
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.

Configure Authentication Context Requests at the SP

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.

Create an authentication context template (see page 138) before you enable authentication context requests or while you are configuring the SP partnership.

Enable Authentication Context Requests at the SP

An SP can request that an IdP return the authentication context in an assertion. Enable that request at the SP->IdP partnership.

Before you begin, we recommend that you create an authentication context template.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Federation, Partnership Federation, Partnerships.
3. Select the SP->IdP partnership you want to edit.
4. Navigate to the Configure AuthnContext step in the partnership wizard.

   The configuration dialog opens.
5. Select the Enable Authentication Context Processing check box.

6. Complete the fields in the dialog.

   Note: 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 (see page 138).
   - 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 10: Delegated Authentication

This section contains the following topics:

Delegated Authentication Overview

One of the configuration decisions for single sign-on is determining how users are authenticated.

SiteMinder offers two authentication choices:

- Local authentication

  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

  SiteMinder uses a third-party web access management (WAM) application that SiteMinder does not protect. The third-party application authenticates any user who requests a protected federated resource then forwards the federated user identity to SiteMinder. After 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 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 SiteMinder at the Asserting Party

SiteMinder can initiate an authentication request at an asserting party. If the request is made to SiteMinder, it is recognized as a delegated authentication request. 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 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 SiteMinder at the asserting party. 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 SiteMinder.

After the third-party WAM system receives the authentication request, it passes the user identity to 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 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 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 SiteMinder.

The methods of passing the user identity are detailed in the following sections.

Cookie Method for Passing User Identity

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 SiteMinder. If authentication begins at 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 Federation Manager 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 Federation Manager 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 Federation Manager 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 Federation Manager 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 SiteMinder supports for password-based encryption:

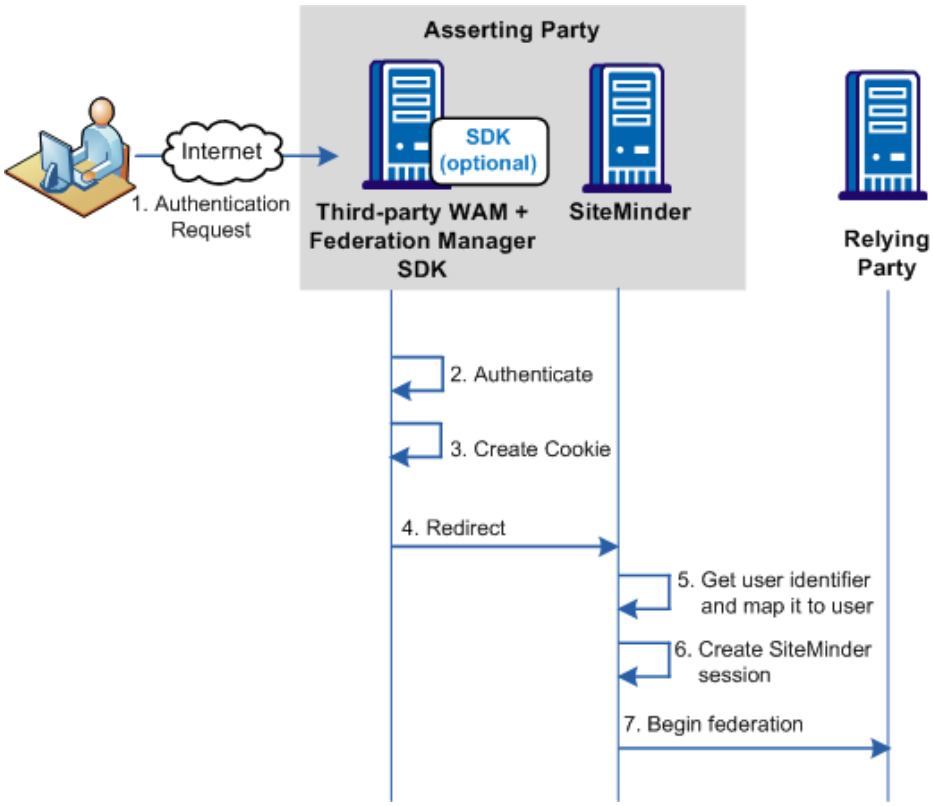
     - PBE/SHA1/AES/CBC/PKCS12PBE-1000-128
     - PBE/SHA1/AES/CBC/PKCS12PBE-1000-192
     - PBE/SHA1/AES/CBC/PKCS12PBE-1000-256
     - PBE/SHA256/AES/CBC/PKCS12PBE-1000-128
     - PBE/SHA256/AES/CBC/PKCS12PBE-1000-192
     - PBE/SHA256/AES/CBC/PKCS12PBE-1000-256
     - PBE/SHA1/3DES_EDE/CBC/PKCS12PBE-1000-3
     - 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 SiteMinder must be in the same cookie domain.
4. The WAM system redirects the browser to SiteMinder.
5. SiteMinder extracts the login ID from the cookie then locates the user in its user directory.
6. SiteMinder creates a 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. SiteMinder is not protecting the WAM application.



Important! To use an SDK-created open-format cookie, the third party must install a Federation Manager SDK. The SDK is a separately installed component from 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 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 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 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.*

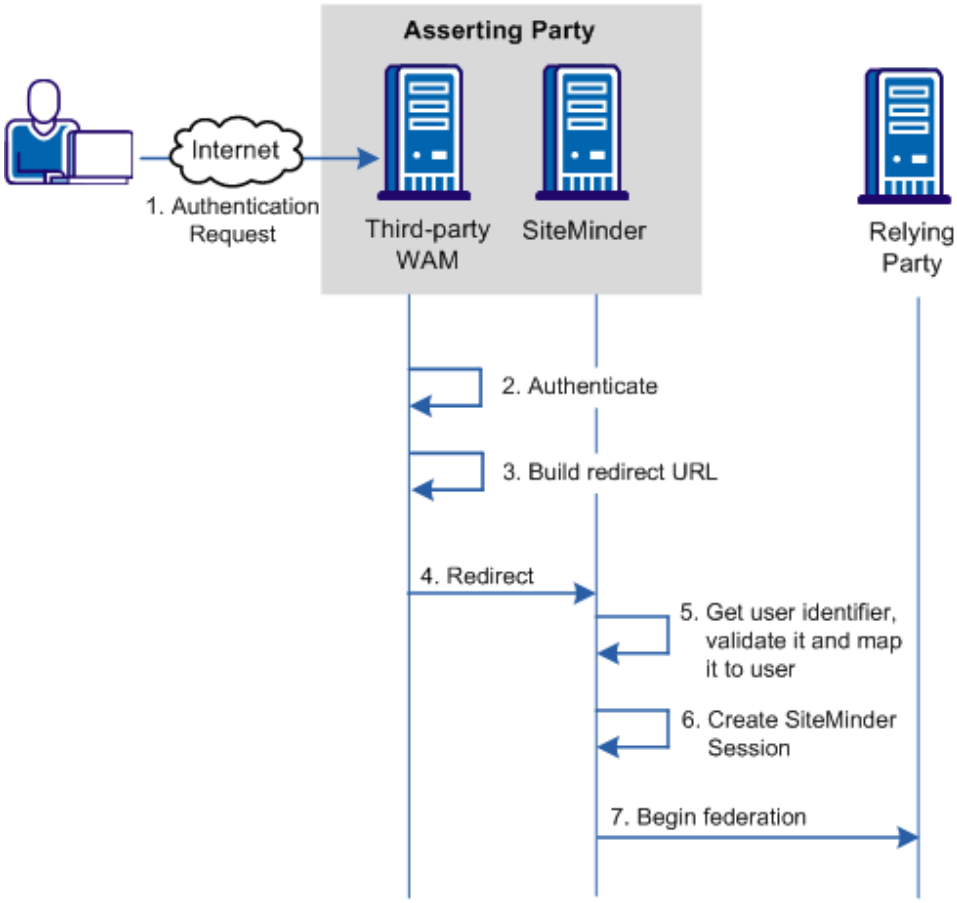
   Important! The LoginID and LoginIDHash parameters are case-sensitive. Be sure to include them in the redirect URL as shown in the example.

   The hashing mechanism allows 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 SiteMinder.
5. 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. 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 Federation Manager 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 tab 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 Federation Manager 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 Federation Manager 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 Federation Manager 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.

   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 tab 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 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 Federation Manager Java SDK to create a legacy cookie or an open-format cookie.
  - For .NET applications, the WAM can use a Federation Manager .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 *Federation Manager Java SDK Guide* or the *Federation Manager .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 SiteMinder asserting party:
  - Encryption Password
  - Open-format Cookie Name
  - Open-format Cookie Encryption Transformation

  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 SiteMinder. This URL has to send the user back to the SiteMinder single sign-on service. The 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 SiteMinder, it must capture and resend any existing query string. The incoming request can have 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 SiteMinder at the asserting party.

Third-party WAM Configuration for Query String Delegated Authentication

A third-party WAM system and 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.

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 SiteMinder configuration at the asserting party.

When SiteMinder retrieves the credentials from the query string, it also combines these values and hashes them. If the hashes are equal, SiteMinder considers the login ID to be valid and continues with the federation request.

Important! The LoginID and LoginIDHash parameters are case-sensitive.

The third-party WAM system must configure its federated application to construct a redirect URL that sends the user back to the SiteMinder Single Sign-on service. Therefore, the 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 SiteMinder, it captures and resends any existing query string. If the incoming request has 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 11: 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, 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 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 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. 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 SiteMinder Asserting Party

The initial authentication of a user at a 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.

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 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 12: Log Files which Aid Troubleshooting

This section contains the following topics:

Federation Web Services Trace Logging

The Web Agent trace logging facility and the Policy Server Profiler enable SiteMinder to monitor the performance of the Web Agent and Policy Server. These logging mechanisms provide comprehensive information about the operation of SiteMinder processes so you can analyze performance and can troubleshoot issues.

For partnership federation, several logging components are available to collect trace messages for federated communication. Trace messages provide detailed information about program operation for tracing, debugging, or both. Trace messages are ordinarily turned off during normal operation. You can enable them to extract in-depth information in addition to the trace message itself. For example, you can look at the FWSTrace.log to see the generated SAML assertion or collect the name of the current user.

The collected trace messages are written to a trace log. The FWSTrace.log is located in the directory *web_agent_home*/log.

You can establish trace logs at the Web Agent and the Policy Server to monitor SiteMinder operation.

FWS Log Messages at the Policy Server

The component that controls the trace messages for federation services at the Policy Server is the Fed_Server component. This component monitors activity for the assertion generator and the SAML authentication scheme. For example, you can view the generated assertion in the smtracedefault.log file.

To configure logging at the Policy Server, use the Policy Server Profiler. The Profiler is available from the Policy Server Management Console. The Profiler is a graphical user interface that lets you specify components for trace logging, which include:

trace configuration file

Defines the components and subcomponents that are included in the file.

trace log file

Specifies the output file for all the logged messages.

The following subcomponents are available for the Fed_Server component:

Configuration

Monitors SAML 2.0 Service Provider configuration activity.

Assertion_Generator

Watches the activity for the SAML 1.x and 2.0 assertion generators.

Auth_Scheme

Monitors the activity of the SAML 1.x or SAML 2.0 authentication schemes.

Saml_Requester

Watches SAML Requester activity

Attribute_Service

Watches the Attribute Service activity

Use the SiteMinder Profiler to Log Trace Messages

The profiler is the Policy Server facility for logging. You can use the profiler to collect trace messages for federation services.

Access the profiler from the Policy Server Management Console.

To configure the profiler

1. Open the Policy Server Management Console.
2. Select the Profiler tab.
3. Select the Enable Profiling check box.
4. In the Configuration File field, click Browse and locate the template that you want to use.

   You can load the default template, smtracedefault.txt, in the directory *policy_server_home*/config, or one of the preconfigured templates in the directory *policy_server_home*/config/profiler_templates.

5. In the Output section, select whether to log data to the Console or to a File or both. If you select a file, specify a path to that file in the Output to File field then select an output format.

   Note: Verify that the log file uses a unique name.

6. Click OK to save your changes.

FWS Log Messages at the Web Agent

The Federation Web Services (FWS) application that is installed with the Web Agent Option Pack, represents the federation client. The component that controls the trace messages and monitors FWS activity is the Fed_Client component.

Within the Fed_Client component, the following sub components are included:

single sign-on

Monitors single sign-on activity.

single logout

Monitors requests for single logout.

discovery profile

Monitors the identity provider discovery profile activity.

administration

Watches administration-related messages.

request

Monitors request and authentication activity.

general

Monitors activity that other subcomponents are not monitoring.

configuration

Monitors SAML 2.0 Service Provider configuration messages.

FWS uses the common tracing facility that the Web Agent uses to log trace messages. The following files are used to set up trace logging:

trace configuration file

Specifies the configuration file that determines which components and events FWS monitors. The default file is FWSTrace.conf.

trace log file

Specifies the output file for all the logged messages. You provide a name and the location for this file in the Web Agent configuration file.

Web Agent Configuration File or Agent Configuration Object

Contains the logging parameters that enable logging and format the log. This file does not define message content.

Configure FWS Trace Logging

To collect trace messages for the Federation Web Services application, configure the FWS trace logging.

Follow these steps:

1. Do one of the following tasks:
   - Make a copy of the default template, FWSTrace.conf and modify the file to include only the data you want to monitor.
   - Copy one of the preconfigured templates and assign a new name to it.

   Note: Do not edit the template directly.
2. Open the LoggerConfig.properties file in the directory *web_agent_home*/affwebservices/WEB-INF/classes, and set the following parameters:
   - Set TracingOn to Yes. This option instructs the trace facility to write messages to a file.
   - Set the TraceFileName parameter to the full path of the trace log file. The default location is in *web_agent_home*/config/FWSTrace.log.
   - Set the TraceConfigFile parameter to the full path of the trace configuration file, either the default template, FWSTrace.conf or another template. Templates can be found at *web_agent_home*/config.
3. Optionally, you can format the trace log file, the file that contains the log output. The following parameters are the Web Agent configuration parameters that dictate the format of the trace log file:
   - TraceRollover
   - TraceSize

- TraceCount
- TraceFormat
- TraceDelim

The LoggerConfig.properties file contains descriptions of all these settings.

Resolving Signature Verification Failures

A malicious user can commit an XML signature wrapping attack by changing the content of a document without invalidating the signature. By default, software controls for the Policy Server and Web Agent Option Pack are set to defend against signature wrapping attacks. However, a third-party product can issue an XML document in a way that does not conform to XML specifications. As a result, the default signature checks can result in a signature verification failure.

Signature verification failures occur for the following reasons:

- A duplicate ID element is in the XML document and the signature references this duplicate ID. Duplicate ID attributes are not permitted.
- The XML signature does not reference the expected parent element, and a signature wrapping vulnerability is logged.

If a federation transaction fails, examine the smtracedefault.log file and the fwstrace.log file for a signature verification failure. These errors can indicate that the received XML document is not conforming to XML standards. As a workaround, you can disable the default Policy Server and Web Agent protection against signature wrapping attacks.

Important! If you disable the protection against signature vulnerabilities, determine another way to protect against these attacks.

To disable the XML signature wrapping checks:

1. Navigate to the xsw.properties file. The file exists in different locations for the Policy Server and the Web Agent.
   - For error messages in the Policy Server smtracedefault.log file, go to *siteminder_home*/config/properties
   - For error messages in the Web Agent fwstrace.log, go to

     *web_agent_option_pack_ home/*affwebservices/web-INF/classes.

     Note: If the web agent option pack is installed on the same system as the web agent, the file resides in the *web_agent_home* directory.

2. Change the following xsw.properties settings to true:
   - DisableXSWCheck=true (Policy Server setting only)
   - DisableUniqueIDCheck=true (Policy Server and Web Agent Option Pack setting)

     Note: The value of the DisableUniqueIDCheck setting must be the same for the Policy Server and the Web Agent Option Pack.
3. Save the file.

Simplify Logging with Trace Configuration Templates

To make the task of collecting tracing data simpler, a series of preconfigured templates are installed with the Policy Server and the Web Agent Option Pack. You can use these templates instead of creating your own trace configuration file to collect the data that gets written to a trace log.

Trace Logging Templates for the IdP and SP

The following templates are available for trace logging related to the Identity Provider and the Service Provider, such as assertion generation or SAML authentication.

Template	Tracing Messages Collected
samlidp_trace.template	Collects messages for Identity Provider activity
samlsp_trace.template	Collects messages for Service Provider activity

Look at each template to see the exact contents. The templates are located in *policy_server_home*/config/profiler_templates.

To use the template

1. Open the Policy Server Management Console.
2. Select the Profiler tab.
3. Select the Enable Profiling check box.
4. In the Configuration File field, click Browse and locate the template that you want to use.

5. In the Output section, select whether to log the data to the Console or to a File or both. If you select a file, specify a path to that file in the Output to File field and select an output format.

   Note: Verify that the log file uses a unique name.

6. Click OK to save your changes.

Service Provider Template Sample

The following text is the samlsp_trace.template file.

```
components: Server/Policy_Server_General, IsProtected/Resource_Protection,
Login_Logout/Authentication, Login_Logout/Policy_Evaluation,
Login_Logout/Active_Expression, Login_Logout/Session_Management,
IsAuthorized/Policy_Evaluation, JavaAPI, Fed_Server/Auth_Scheme,
Fed_Server/Configuration
data: Date, Time, Tid, TransactionID, SrcFile, Function, Domain, Resource, Action,
User, Message
```

For legacy federation, it includes the Fed_Server component along with the subcomponents Auth_Scheme and Configuration.

The data fields that indicate the required contents of each message are:

Date, Time, Tid, TransactionId, SrcFile, Function, Domain, Resource, Action User, and Message.

Identity Provider Profiler Sample

At the Identity Provider, the Profiler tab of the Policy Server Management Console specifies a template in the Configuration File field. The following text is a sample entry for the Configuration File field:

c:\program files\ca\siteminder\config\profile_templates\samlidp_template.trace

For more information about using the Profiler, see the *Policy Server Administration Guide*.

Trace Logging Templates for FWS

The following templates are available for Federation Web Services:

Template	Tracing Messages Collected
WebAgentTrace.conf	Default template. Collects data that you specify.
FWS_SSOTrace.conf	Collects single sign-on messages

Template	Tracing Messages Collected
FWS_SLOTrace.conf	Collects single logout messages
FWS_IPDTrace.conf	Collects Identity Provider Discovery Profile messages

All these templates include the Fed_Client component and subcomponents for the specific data being tracked. Look at each template to see the exact contents. The templates are located in *web_agent_home*/config.

To use a template for trace logging

1. Make a copy of the template you want to use and rename the copy.

   Note: Do not edit the template directly.

2. Open the Agent configuration file or Agent configuration Object.
3. Set the TraceFile parameter to Yes.
4. Set the TraceFileName parameter to the full path to the trace log file. This file contains the log output.
5. Set the TraceConfigFile parameter to the full path to the newly named template file.
6. Format the trace log file. The following parameters are the Web Agent configuration parameters that dictate the format of the trace log file:
   - TraceAppend
   - TraceFormat
   - TraceDelimiter
   - TraceFileSize
   - LogLocalTime

   For descriptions of each logging parameter, see the *Web Agent Configuration Guide*.

Note: Web Agents on IIS and Apache 2.0 servers do not support dynamic configuration of log parameters that are set locally in the Agent configuration file. Consequently, when you modify a parameter, the change takes effect only after the Agent is restarted. If you configure the log parameters in an Agent configuration object, these log settings can be stored and updated dynamically.

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
```

Update FWS Data in the Logs

If you modify any part of the federation configuration, flush the Federation Web Services cache for the changes to appear in the trace logs.

Note: A brief delay can occur from when the changes are made and when Federation Web Services receives the information.

To flush the cache

1. Log in to the Administrative UI.
2. Select Administration, Policy Server, Cache Management.

   The Cache Management page displays.
3. Click Flush All in the All Caches section of the page.
4. Click Close

Federation Database Objects Trace

Enable XPS validation and federation object tracing to monitor federation database activities. SiteMinder logs these activities to the smps.log file, in the directory *siteminder_home*\log.

Follow these steps:

1. Open a command window.
2. Type XPSConfig.

   Type the command as it is shown here. The command is case-sensitive.

   The Products Menu displays.
3. Enter XPS.

   The Parameters Menu displays.
4. Enter the number for the LogValidationWarnings parameter.

5. Enter c to change the value to true. With a Boolean value, XPSConfig automatically changes the value to the opposite of the current setting.

   When set to true, the parameter enables tracing for XPS Validation warnings.

6. Enter the number for the Trace parameter.
7. Enter c to change the value to true.

   When set to true, the parameter enables XPS trace debugging.

8. Enter q until you return to the Product Menu.
9. Enter FED.

   The FED parameters menu display.

10. Enter the number for the TRACE parameter.
11. Enter c to enable it.

    When set to true, the TRACE parameter enables tracing for federation XPS objects.

12. Enter q to return to the Parameters Menu. If you are done modifying parameters, keep entering q until you exit XPSConfig.

    Changes made in XPSConfig are not recognized until you exit the XPSConfig tool. Where noted, some changes require that you restart services.

13. Restart the Policy Server.

Appendix A: Open Format Cookie Details

This section contains the following topics:

Contents of the Open Format Cookie

The federation open format cookie lets applications assert user attributes to SiteMinder and consume user attributes that SiteMinder encapsulates. The open format cookie has the following general characteristics:

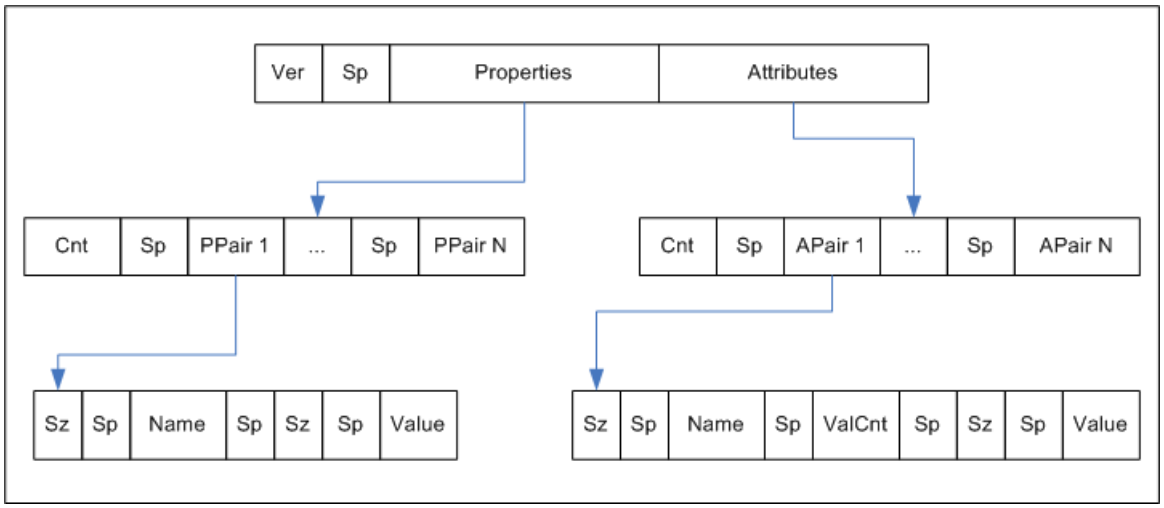
- The cookie is accessible by applications written in any programming language.
- The cookie content consists of a string of UTF-8 bytes, which supports international character sets.
- The combined size in UTF-8 bytes of each name/value pair precedes the name/value pair.
- Space characters are added 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 Federation Manager 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 Federation Manager 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

Chapter 13: Secure Proxy Engine Logs for Federation

Partnership-based federation contains a secure proxy engine that forwards traffic to backend servers. The secure proxy engine includes the following components:

- Apache Web Server

  Acts as the HTTP listener, handling HTTP traffic for incoming requests, and can handle HTTPS traffic, once properly configured.

- Tomcat server

  Provides a servlet container for the operation of the UI. The Apache web server communicates to the Tomcat server through a Tomcat connector named mod_jk.

You can supply CA Support with log files related to these components to troubleshoot problems in your partnership federation environment.

Two Apache logs that aid partnership federation troubleshooting are:

mod_jk.log

mod_jk.log is enabled by default with the product. After the first contact with the federation server, information begins logging to this file. The mod_jk.log file is located in *federation_mgr_home*\logs\fws.

To modify this log file:

a. Navigate to federation_mgr_home\secure-proxy\httpd\conf

b. Open the httpd.conf file.

c. Change the following lines

```
JkLogFile "federation_mgr_home/logs/fws/mod_jk.log"
JkLogLevel error
```

To disable the mod_jk.log, comment out or remove these lines from the file.

httpclient.log

For debug purposes only, you can enable the httpclient.log. The httpclient.log file is located in *federation_mgr_home*\secure-proxy\proxy-engine\logs.

To modify this log file:

a. Navigate to *federation_mgr_home*\secure-proxy\proxy-engine\conf.

b. Open the server.conf file

c. Change the following line:

```
httpclientlog="yes"
```

To modify the location of the httpclient.log file and the log level, edit the httpclientlogging.properties file. This file is in the directory *federation_mgr_home*\secure-proxy\Tomcat\properties.

Appendix B: 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

SiteMinder uses the following algorithms for partnership signing options.

Encryption Key Algorithms

RSA-V15, RSA-OEAP

Encryption Block Algorithms

3DES, AES-128, AES-256

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, 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 Federation Manager 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

Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U