SAML v1.x Guide

ForgeRock Access Management 5

ForgeRock AS
201 Mission St, Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide to working with SAML v1.x. ForgeRock# Access Management provides authentication, authorization, entitlement and federation software.

Table of Contents

Preface iv
1. Introducing SAML v1.x Single Sign-On 1
    1.1. About SAML v1.x 1
2. Implementing SAML v1.x 4
    2.1. Gathering Configuration Information 4
    2.2. Preparing To Secure SAML v1.x Communications 6
    2.3. Configuring SAML v1.x For Your Site 7
    2.4. Configuring SAML v1.x Trusted Partners 9
    2.5. Testing SAML v1.x Web SSO 11
3. Reference 15
    3.1. Common Federation Configuration 15
A. Getting Support 21
    A.1. Accessing Documentation Online 21
    A.2. Joining the ForgeRock Community 22
    A.3. Getting Support and Contacting ForgeRock 22
Glossary 23

Preface

This guide covers concepts, configuration, and usage procedures for working with Security Assertion Markup Language (SAML) v1.x features provided by ForgeRock Access Management. ForgeRock Access Management supports SAML version 1.0 and 1.1.

This guide is written for anyone using ForgeRock Access Management for SAML v1.x identity and service providers.

About ForgeRock Identity Platform™ Software

ForgeRock Identity Platform™ is the only offering for access management, identity management, user-managed access, directory services, and an identity gateway, designed and built as a single, unified platform.

The platform includes the following components that extend what is available in open source projects to provide fully featured, enterprise-ready software:

- ForgeRock Access Management (AM)
- ForgeRock Identity Management (IDM)
- ForgeRock Directory Services (DS)
- ForgeRock Identity Gateway (IG)

Chapter 1
Introducing SAML v1.x Single Sign-On

This chapter describes OpenAM's support for the SAML v1.x framework for exchanging security data.

SAML v1.x is an XML- and SOAP-based framework that allows online trusted partners to exchange security information. In particular, SAML v1.x defines mechanisms for browser based web single sign-on (SSO) across independent organizations that work together to permit SSO for access to resources.

Important

Although not strictly compatible with SAML v1.x, SAML v2.0 extends SAML v1.x to several additional use cases and also clarifies how partners share metadata with each other. Unless you are integrating with an existing SAML v1.x deployment consider using SAML v2.0, or an alternative, such as OAuth 2.0 or OpenID Connect 1.0, instead.

See the following for more information: SAML v2.0 Guide, OAuth 2.0 Guide, and Chapter 1, "*Introducing OpenID Connect 1.0*" in the *OpenID Connect 1.0 Guide*.

OpenAM's support for SAML 1.x requires stateful sessions. Be sure that OpenAM is configured for stateful sessions—the default configuration—before attempting to use SAML 1.x functionality in OpenAM.

1.1. About SAML v1.x

SAML v1.x was defined in response to several technical problems:

- Web SSO solutions often use SSO session cookies. Browsers do not return cookies across domains. For instance, browsers do not return cookies set by servers in the example.com domain to servers in the example.net domain. SAML v1.x works around this limitation of HTTP cookies.
- Before SAML v1.x was defined, there were already proprietary SSO solutions, but the solutions did not interoperate well across domains.

  SAML v1.x specifies a standard, cross-domain, interoperable SSO mechanism that works together with proprietary SSO services in a particular domain.
- Before SAML v1.x was defined, there was not an easy way to communicate security attributes across organization boundaries.

  SAML v1.x simplifies the communication of security attributes between different organizations.

In SAML v1.x, business partners can play two roles. The *asserting party*, also known as the SAML authority and whose domain is the Source site, authenticates users and asserts information about them. The *relying party*, whose domain is known as the Destination site, consumes assertions and uses information from the assertion to decide whether to allows access to resources.

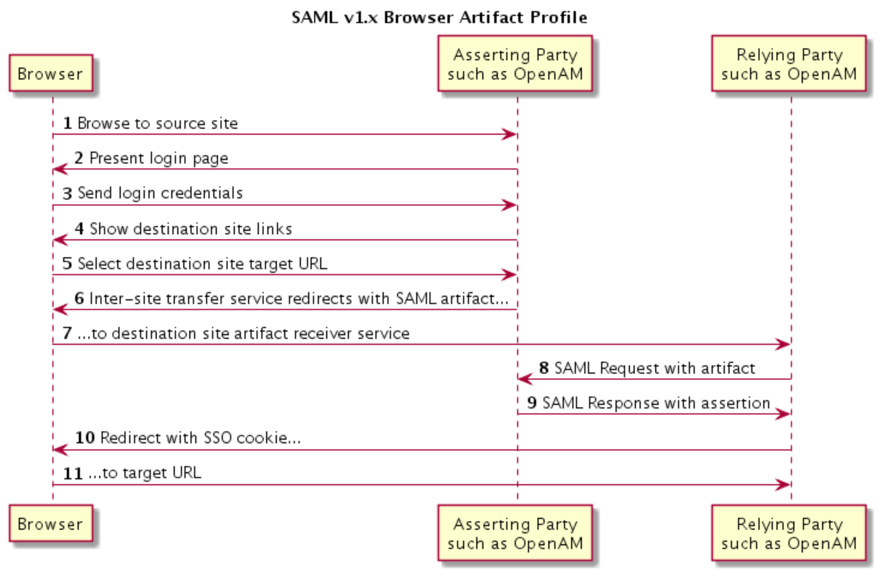
In the Web Browser SSO Profiles for SAML v1.x, the user generally starts by authenticating with the asserting party and then selecting a relying party link to browse. Alternatively, the "Destination-Site-First" scenario can start with the user browsing to the relying party's site and being redirected to the asserting party's site to authenticate.

The SAML v1.x *Inter-site Transfer Service* is a service that redirects the authenticated user from the asserting party's site to the appropriate service on the relying party's site. The Inter-site Transfer Service also handles artifact and redirect generation. How this service transfers the user to the relying party's site depends on how the asserting party and the relying party exchange messages.

The asserting party and relying party can exchange messages either by reference, where the asserting party sends an *artifact* (a base64-encoded reference to the assertion) as a query string parameter value, or by value, where the asserting party directs the user's browser to HTTP POST the assertion to the relying party.

When the asserting party and relying party use artifacts, the Inter-site Transfer Service redirects the user's browser to the relying party's Artifact Receiver Service with the artifact as the value of a query string parameter. The relying party retrieves the artifact from the query string, and then sends a SAML Request to the Responder Service at the asserting party. The asserting party replies with a SAML Response containing one or more assertions.

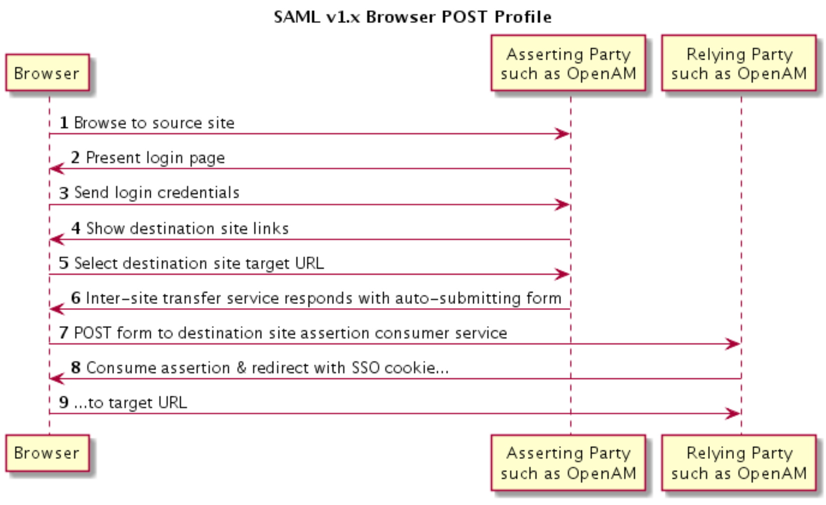
Figure 1.1. SAML v1.x Web SSO Browser Artifact Profile


SAML v1.x Browser Artifact Profile
Browser
Asserting Party such as OpenAM
Relying Party such as OpenAM
1 Browse to source site
2 Present login page
3 Send login credentials
4 Show destination site links
5 Select destination site target URL
6 Inter-site transfer service redirects with SAML artifact...
7 ...to destination site artifact receiver service
8 SAML Request with artifact
9 SAML Response with assertion
10 Redirect with SSO cookie...
11 ...to target URL
Browser
Asserting Party such as OpenAM
Relying Party such as OpenAM

See section 4.1.1 of the SAML v1.1 technical overview for more detail.

When the assertion is sent using the Browser/POST Profile, the Inter-site Transfer Service responds to the user's browser with an auto-submitting form containing the SAML response. The browser then submits the SAML response as form data by HTTP POST to the relying party's Assertion Consumer Service. The relying party's Assertion Consumer Service then processes the assertion.

Figure 1.2. SAML v1.x Web SSO Browser POST Profile


SAML v1.x Browser POST Profile
Browser
Asserting Party such as OpenAM
Relying Party such as OpenAM
1 Browse to source site
2 Present login page
3 Send login credentials
4 Show destination site links
5 Select destination site target URL
6 Inter-site transfer service responds with auto-submitting form
7 POST form to destination site assertion consumer service
8 Consume assertion & redirect with SSO cookie...
9 ...to target URL
Browser
Asserting Party such as OpenAM
Relying Party such as OpenAM

See section 4.2.1 of the SAML v1.1 technical overview for more detail.

The Assertion Consumer Service at the relying party validates the digital signature on the SAML response, and redirects the browser to the target URL of the resource that the user is attempting to access. The server providing that resource uses the relying party's authorization decision capabilities to establish whether the user can access the resource. If so, the resource is returned to the user's browser. If the relying party is using OpenAM, for example, then the relying party sets an OpenAM SSO token based on the SAML response, and this token is used to track the user's session for authorization.

Organizations working together to achieve SAML v1.x web SSO are called *trusted partners* in this context. Trusted partners agree on which services they provide, which web SSO profiles they implement, and how information is exchanged in the assertions, including profile attribute values. Once the trusted partners have reached agreement on how they interact, you can collect information about your partners' configurations and configure OpenAM to match your organization's part of the agreement.

Chapter 2
Implementing SAML v1.x

This chapter covers implementation of OpenAM's SAML v1.x component and covers the following topics:

• Section 2.1, "Gathering Configuration Information"
• Section 2.2, "Preparing To Secure SAML v1.x Communications"
• Section 2.3, "Configuring SAML v1.x For Your Site"
• Section 2.4, "Configuring SAML v1.x Trusted Partners"
• Section 2.5, "Testing SAML v1.x Web SSO"

2.1. Gathering Configuration Information

Before you can configure OpenAM to allow web SSO with trusted partners, you must first gather information about the agreement itself, as well as information for your site and for your partners sites.

This section lists the data that you must collect:

- SAML protocol version to use (1.1 or 1.0; default: 1.1)
- Assertion version to use (1.1 or 1.0; default: 1.1)
- Which trusted partners play which roles (asserting party, relying party)
- Domain names of partner sites (for example, `example.com`, `example.net`)
- Whether assertions are exchanged by SAML artifact or by HTTP POST

  If assertions are exchanged by artifact, also gather this information:

  - SAML artifact parameter name (default: `SAMLart`)
  - Artifact timeout
  - URL to the relying party endpoint that receives the artifact (for example, `https://rp.example.com/openam/SAMLAwareServlet`)
  - Relying party hosts that consume artifacts (by IP addresses, DNS names, or certificate aliases)
  - URL to the asserting party endpoint that responds to SAML requests (for example, `https://ap.example.net/openam/SAMLSOAPReceiver`)

  - Authentication credentials to connect to the asserting party endpoint, if any (for example, the username and password for HTTP Basic authentication)
  - Asserting party signing certificate alias

  If assertions are exchanged by HTTP POST, also gather this information:

  - URL to the relying party endpoint that consumes the form data in the POST assertion (for example, `https://rp.example.com/openam/SAMLPOSTProfileServlet`)
  - Asserting party host:port issuing assertions
  - Asserting party signing certificate alias

- Whether the relying party sends SOAP query requests to the asserting party, for example, to get authorization decisions

  If the relying party queries the asserting party, also gather this information:

  - Relying party hosts that consume artifacts (by IP addresses, DNS names, or certificate aliases)
  - How to get SSO information, and to map partner actions to authorization decisions
  - Asserting party host:port issuing assertions
  - Asserting party signing certificate alias

- Target specifier parameter name (default: `TARGET`)
- Assertion timeout
- Whether to digitally sign assertions, requests, responses
- Partners' public key certificates used for HTTPS
- Partners' public key certificates used for message signing (unless included on the `KeyInfo` element of signed messages)
- Partners' Site IDs (base64-encoded ID, for example, `XARFfsIAXeLX8BEWNIJg9Q8r0PE=`)
- What NameID formats are used to exchange names (for example, `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`)
- How attributes map from an assertion to an OpenAM profile (for example, `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress|EmailAddress=mail`)

For more information about your own public key certificates, see Section 2.2, "Preparing To Secure SAML v1.x Communications".

For your own Site ID, see the following procedure.

Procedure 2.1. To Generate a Site Identifier for a Site

Trusted partners should ask you for a Site ID. OpenAM generates a SAML v1.x Site ID value at configuration time. This Site ID value corresponds to the server. To find this in the AM console, see Federation > SAML 1.x Configuration > Local Site Properties > Site Identifiers, and then click your server URL.

If you have multiple servers in an OpenAM site set up behind a load balancer, you can generate a Site ID, and then use it for all the servers in your site.

- Generate a Site ID for your site, using the primary site URL.

  This example is for an asserting party where the site load balancer host is `ap.example.net`. The command is bundled with OpenAM server, shown with lines folded to fit on the printed page:

  ```
  $ cd /path/to/tomcat/webapps/openam/WEB-INF/lib/
  $ java \
       -cp forgerock-util-21.0.0.jar:openam-shared-14.0.0.jar:\
       openam-federation-library-14.0.0.jar com.sun.identity.saml.common.SAMLSiteID \
       https://ap.example.net/openam
       9BAg4UmVS6IbjccsSj9gAFYGO9Y=
  ```

2.2. Preparing To Secure SAML v1.x Communications

SAML communications are secured using Public Key Infrastructure (PKI). Communications should be protected over the network by HTTPS, and relying parties requesting assertions should use SSL or TLS mutual authentication to identify each other, meaning they should be able to trust each others' certificates. Furthermore, when an asserting party works through the user's browser to post an assertion to the relying party, then the asserting party must digitally sign the SAML response.

A certificate can be trusted when the signer's certificate is trusted, or when the certificate itself is trusted. Trusted partners must either use public key certificates signed by a well-known Certificate Authority (CA), or share their self-signed or private CA signing certificates.

Procedure 2.2. To Configure Keys For Protecting SAML v1.x Communications

1. See Chapter 5, "*Setting Up Keys and Keystores*" in the *Setup and Maintenance Guide* for instructions on handling your own key pairs.

   For specific instructions on changing signing keys, see Procedure 5.3, "To Change Default test Signing Key" in the *Setup and Maintenance Guide*.

2. If necessary, share signing certificates with trusted partners.

3. Import public key certificates shared by trusted partners into your OpenAM keystore.

2.3. Configuring SAML v1.x For Your Site

After you have gathered configuration information and prepared to secure SAML v1.x communications you can configure SAML v1.x for your site.

Tip

When you enter SAML v1.x configuration data, The AM console escapes these special characters by default: `& < > " ' /`. If you have already escaped these characters in the data that you plan to enter in the AM console, set the value of the `com.sun.identity.saml.escapeattributevalue` property to `false` under Configure > Server Defaults > Advanced, and then restart OpenAM or the container in which it runs to prevent The AM console from escaping the characters for you.

- Procedure 2.3, "To Configure Asserting Party Local Site Properties"
- Procedure 2.4, "To Configure Relying Party Local Site Properties"

Procedure 2.3. To Configure Asserting Party Local Site Properties

Using the configuration information you have gathered complete the following steps:

1. Log in to the AM console as administrator, amadmin, navigate to Realms > *`Realm Name`* > Applications > SAML > SAML 1.x Configuration, and then click Local Site Properties.

2. If the target specifier query string parameter is something other than the standard default TARGET, set it in the Target Specifier field.

3. If instead of the default server Site Identifier, you use a Site Identifier for the OpenAM Site, click New in the Site Identifiers table, and then add the information for the OpenAM Site, including the Site ID that you generated.

4. Target URLs let you configure URLs for which HTTP POST is always used.

   When the TARGET specified matches a URL in the Target URLs list, then the asserting party sends the response to the relying party by HTTP POST of an auto-submitting form returned to the browser.

5. If necessary, set the Default Protocol Version.

6. In the Assertion section, change the values if necessary.

   Remove Assertion: Yes means that assertions are deleted from memory after they are used, rather than deleted only when they expire.

7. In the Artifact section, change the values if necessary.

8. In the Signing section, for an asserting party using the HTTP POST profile, check at least Sign SAML Assertion.

   By default OpenAM signs messages using the certificate with alias test.

Check other options as required by your trusted partners.

9. In the Attribute Query section, if relying parties issue attribute queries, then set the default list of profile attributes to return.
10. In the NameID Format section, map SAML NameID formats to local OpenAM user profile attributes.

    This allows OpenAM to map a remote user to a local user profile.
11. In the Attribute Map section, if the parties exchange attributes, then map the SAML attributes requested by relying parties to local OpenAM user profile attributes.
12. Save your work.

Procedure 2.4. To Configure Relying Party Local Site Properties

Using the configuration information you have gathered complete the following steps.

1. Log in to the AM console as administrator, amadmin, navigate to Realms > *`Realm Name`* > Applications > SAML > SAML 1.x Configuration, and then click Local Site Properties.
2. If the target specifier query string parameter is something other than the standard default TARGET, set it in the Target Specifier field.
3. If instead of the default server Site Identifier, you use a Site Identifier for the OpenAM Site, click New in the Site Identifiers table, and then add the information for the OpenAM Site, including the Site ID that you generated.
4. Ignore the Target URLs table for a relying party.
5. If necessary, set the Default Protocol Version.
6. In the Assertion section, change the values if necessary.
7. In the Artifact section, change the values if necessary.
8. Ignore the Signing section for relying parties, unless trusted partners require that your site signs SAML requests.

   By default OpenAM signs messages using the certificate with alias `test`.
9. Ignore the Attribute Query section for relying parties.
10. In the NameID Format section, map SAML NameID formats to local OpenAM user profile attributes.

    This allows OpenAM to map a remote user to a local user profile when not all the partners are using OpenAM user IDs.

11. In the Attribute Map section, if the parties exchange attributes, then map the SAML attributes requested by relying parties to local OpenAM user profile attributes.

12. Save your work.

2.4. Configuring SAML v1.x Trusted Partners

After you have gathered configuration information and if necessary imported public key certificates from trusted partners you can configure SAML v1.x information for the partners:

- Procedure 2.5, "To Configure a Trusted Relying Party"
- Procedure 2.6, "To Configure a Trusted Asserting Party"

Procedure 2.5. To Configure a Trusted Relying Party

The AM console refers to the relying party as the Destination, because the relying party's site is the destination site:

1. Log in to the AM console as administrator, amadmin, navigate to Realms > *`Realm Name`* > Applications > SAML > SAML 1.x Configuration, and then click New in the Trusted Partners table.

2. Under Destination, select the SAML profiles used with the relying party.

3. In the Common Settings section, set at least a name for the partner configuration, enter the partner's Site ID as the Source ID, and specify the fully qualified domain, optionally with the port number, of the relying party in the Target field. The value in the target field is matched to TARGET parameter values, so it should correspond to the real domain (and optionally port number) in the URLs of resources to access at the relying party's site.

   Optionally set a custom site attribute mapper, a custom name identifier mapper, and the SAML Version to use with the partner.

   You must also set one or more values in the host list for the partner to identify all hosts from the partner site that can send requests. OpenAM rejects SAML requests from hosts not specified in this list.

4. In the Destination section, if the SAML Artifact profile is used with the relying party, set the SAML URL to the relying party's endpoint that receives the artifact and contacts your asserting party.

   If the SAML POST profile is used with the relying party, set the Post URL to the relying party's endpoint that consumes the assertion in the HTTP POST form data and redirects the user's browser to the target at the relying party's site.

   If the relying party makes SAML SOAP query requests, optionally set custom attribute or action mappers.

If the relying party signs requests, then either requests include the certificate for the signing key in the KeyInfo element, or OpenAM must find the signing certificate elsewhere. If the relying party provides the signing certificate separately, import the signing certificate into OpenAM's default keystore file, and set the alias for the signing certificate here in the configuration. For more information about OpenAM's default keystore, see Chapter 5, "*Setting Up Keys and Keystores*" in the *Setup and Maintenance Guide*.

Set the issuer to a host:port combination corresponding to the relying party server issuing the requests.

5. Save your work.

Procedure 2.6. To Configure a Trusted Asserting Party

The AM console refers to the asserting party as the Source, because the asserting party's site is the source site:

1. Log in to the AM console as administrator, amadmin, navigate to Realms > *`Realm Name`* > Applications > SAML > SAML 1.x Configuration, and then click New in the Trusted Partners table.

2. Under Source, select the SAML profiles used with the asserting party.

3. In the Common Settings section, set at least a name for the partner configuration and enter the partner's Site ID as the Source ID.

   Optionally set a custom account mapper. By default OpenAM maps accounts based on the NameID format configuration for your site.

   If the asserting party signs assertions (or other messages) and you have imported the signing certificate into OpenAM's keystore (also used as a trust store), then enter the signing certificate alias. If instead the asserting party includes the signing certificate in the KeyInfo element of signed messages, then you can leave the alias blank.

4. In the Source section, if the SAML Artifact profile is used with the asserting party, set the SOAP URL to the asserting party endpoint that responds to requests such as `https://ap.example.net/openam/SAMLSOAPReceiver`.

   If the asserting party requires authentication to the SOAP URL, then configure the settings appropriately.

   If the SOAP URL is accessed over HTTP, choose None or Basic. If the SOAP URL is accessed over HTTPS, choose SSL/TLS or SSL/TLS with Basic.

   Basic means HTTP Basic authentication (with username and password). For HTTP Basic authentication, the authentication at this level is performed by the application server container, not OpenAM. Therefore if the asserting party runs OpenAM and wants to enforce HTTP Basic authentication, the asserting party administrator must set up the container to handle HTTP Basic authentication for the SOAP URL.

Set the SAML Version as necessary.

If the SAML POST profile is used with the asserting party, set the Issuer to the issuer name, such as a host:port combination.

5. Save your work.

2.5. Testing SAML v1.x Web SSO

You can try SAML v1.x Web SSO using OpenAM by following the procedures in this section:

- Procedure 2.7, "To Prepare the Servers"
- Procedure 2.8, "To Prepare to Test the Asserting Party"
- Procedure 2.9, "To Prepare to Test the Relying Party"
- Procedure 2.10, "To Try SAML v1.x Web SSO"

Procedure 2.7. To Prepare the Servers

1. Install two separate servers, one to act as asserting party, the other to act as relying party.

   How you do this in practice is up to you.

   You can, for example, set up two separate OpenAM servers on a single host by adding aliases for the hosts in your hosts file, and by using separate containers that listen on different ports.

   For example, if your host is a laptop, you can add the aliases to the loopback address as in the following example line from an `/etc/hosts` file.

   ```
   127.0.0.1    localhost ap.example.net rp.example.com
   ```

   Then, run one application server to listen on port 8080, and another to listen on port 9080.

   Deploy and configure OpenAM server with the default configuration at `http://ap.example.net:8080/ap` for the asserting party and at `http://rp.example.com:9080/rp` for the relying party. This allows you to use the default configuration for both servers.

   See the Installation Guide for instructions.

   The procedures in this section use those example URLs to represent the OpenAM servers acting as asserting and relying parties.

2. On the asserting party server, login to the AM console as administrator, navigate to Realms > *Realm Name* > Applications > SAML > SAML 1.x Configuration, and then click Local Site Properties.

   Click the server's instance ID in the Site Identifiers table.

   Record the asserting party Site ID for later use.

3. On the relying party server, login to the AM console as administrator, navigate to Realms > *Realm Name* > Applications > SAML > SAML 1.x Configuration, and then click Local Site Properties.

   Click the server's instance ID in the Site Identifiers table.

   Record the asserting party Site ID for later use.

Procedure 2.8. To Prepare to Test the Asserting Party

Follow these steps to configure the asserting party OpenAM server:

1. Log in to the AM console as administrator, navigate to Realms > *Realm Name* > Applications > SAML > SAML 1.x Configuration, and then click Local Site Properties.
2. On the Local Site Properties page for the asserting party server, select Sign SAML Response.

   The asserting party thus signs SAML responses with the private key for the default test certificate.
3. Save your work, and then click Back to Federation.
4. Click New in the Trusted Partners table to add the relying party as a trusted partner.
5. In the Destination area of the Select trusted partner type and profile page, select Artifact and Post (not SOAP Query), and then click Next.
6. Apply the following settings, adjusted for the host names you use.

   If a field is not mentioned, accept the defaults.

   Under Common Settings, use these settings:

   Name: rp.example.com:9080
   Source ID: relying party Site ID that you recorded
   Target: rp.example.com:9080

   Under Destination > Artifact, use these settings:

   SOAP URL: http://rp.example.com:9080/rp/SAMLAwareServlet
   Host List: rp.example.com

   Under Source > Post, set Post URL: http://rp.example.com:9080/rp/SAMLPOSTProfileServlet
7. Click Finish to save your work.

Procedure 2.9. To Prepare to Test the Relying Party

Follow these steps to configure the relying party OpenAM server:

1. Log in to the AM console as administrator, navigate to Realms > *`Realm Name`* > Applications > SAML > SAML 1.x Configuration, and then click New in the Trusted Partners table to add the asserting party as a trusted partner.
2. In the Source area of the Select trusted partner type and profile page, select Artifact and Post, and then click Next.
3. Apply the following settings, adjusted for the host names you use.

   If a field is not mentioned, accept the defaults.

   Under Common Settings, use these settings:

   Name: ap.example.net:8080
   Source ID: asserting party Site ID that you recorded
   Signing Certificate Alias: test

   Under Source > Artifact, set SOAP URL: http://ap.example.net:8080/ap/SAMLSOAPReceiver

   Under Source > Post, set Issuer: ap.example.net:8080

   Click Finish to save your work.

Procedure 2.10. To Try SAML v1.x Web SSO

Once you have successfully configured both parties, try SAML v1.x Web SSO:

1. Log out of The AM console on both servers.
2. Try Web SSO using the SAML Artifact profile.
   a. Simulate the OpenAM administrator browsing the asserting party's site, and selecting a link to the AM console on the relying party's site.

      The URL to simulate this action is something like `http://ap.example.net:8080/ap/SAMLAwareServlet?TARGET=http://rp.example.com:9080/rp`.

      OpenAM requires that you authenticate.
   b. Login as OpenAM demo user, `demo` with default password `changeit`, on the asserting party server.
   c. Notice that you are redirected to the AM console on the relying party server, and that you are successfully logged in as the demo user.
   d. Log out of The AM console on both servers.
3. Try Web SSO using the SAML HTTP POST profile:
   a. Simulate the OpenAM administrator browsing the asserting party's site, and selecting a link to the AM console on the relying party's site.

The URL to simulate this action is something like `http://ap.example.net:8080/ap/SAMLPOSTProfileServlet?TARGET=http://rp.example.com:9080/rp`.

OpenAM requires that you authenticate.

b. Login as OpenAM administrator, `amadmin`, on the asserting party server.

c. Notice that you are redirected to the AM console on the relying party server, and that you are successfully logged in as `amadmin`.

Chapter 3 Reference

This reference section covers common configuration settings for federation, some of which apply to OpenAM's SAML v1.x support.

3.1. Common Federation Configuration

ssoadm service name: `federation/common`

3.1.1. General Configuration

The following settings appear on the **General Configuration** tab:

Maximum allowed content length

The maximum content length allowed in federation communications, in bytes.

Default value: `20480`

ssoadm attribute: `maxContentLength`

Check presence of certificates

Enable checking of certificates against local copy

Whether to verify that the partner's signing certificate included in the Federation XML document is the same as the one stored in the said partner's meta data.

The possible values for this property are:

```
off
on
```

Default value: `on`

ssoadm attribute: `certificateChecking`

SAML Error Page URL

OpenAM redirects users here when an error occurs in the SAML2 engine.

Both relative and absolute URLs are supported. Users are redirected to an absolute URL using the configured HTTP Binding whereas relative URLs are displayed within the request.

Default value: `/saml2/jsp/saml2error.jsp`

ssoadm attribute: `samlErrorPageUrl`

SAML Error Page HTTP Binding

The possible values are HTTP-Redirect or HTTP-POST.

Default value: `HTTP-POST`

ssoadm attribute: `samlErrorPageHttpBinding`

3.1.2. Implementation Classes

The following settings appear on the **Implementation Classes** tab:

Datastore SPI implementation class

The Federation system uses this class to get/set user profile attributes.

The default implementation uses the Identity repository APIs to access user profile attributes. A custom implementation must implement the `com.sun.identity.plugin.datastore.DataStoreProvider` interface.

Default value: `com.sun.identity.plugin.datastore.impl.IdRepoDataStoreProvider`

ssoadm attribute: `datastoreClass`

ConfigurationInstance SPI implementation class

The Federation system uses this class to fetch service configuration.

The default implementation uses the SMS APIs to access service configuration. A custom implementation must implement the `com.sun.identity.plugin.configuration.ConfigurationInstance` interface.

Default value: `com.sun.identity.plugin.configuration.impl.ConfigurationInstanceImpl`

ssoadm attribute: `configurationClass`

Logger SPI implementation class

The Federation system uses this class to record log entries.

The default implementation uses the Logging APIs to record log entries. A custom implementation must implement the `com.sun.identity.plugin.log.Logger` interface.

Default value: `com.sun.identity.plugin.log.impl.LogProvider`

ssoadm attribute: `loggerClass`

SessionProvider SPI implementation class

The Federation system uses this class to interface with the session service.

The default implementation uses the standard authentication and SSO APIs to access the session service. A custom implementation must implement the `com.sun.identity.plugin.session.SessionProvider` interface.

Default value: `com.sun.identity.plugin.session.impl.FMSessionProvider`

ssoadm attribute: `sessionProviderClass`

PasswordDecoder SPI implementation class

The Federation system uses this class to decode password encoded by OpenAM.

The default implementation uses the internal OpenAM decryption API to decode passwords. A custom implementation must implement the `com.sun.identity.saml.xmlsig.PasswordDecoder` interface.

Default value: `com.sun.identity.saml.xmlsig.FMPasswordDecoder`

ssoadm attribute: `passwordDecoderClass`

SignatureProvider SPI implementation class

The Federation system uses this class to digitally sign SAML documents.

The default implementation uses the XERCES APIs to sign the documents. A custom implementation must implement the `com.sun.identity.saml.xmlsig.SignatureProvider` interface.

Default value: `com.sun.identity.saml.xmlsig.AMSignatureProvider`

ssoadm attribute: `signatureProviderClass`

KeyProvider SPI implementation class

The Federation system uses this class to provide access to the underlying Java keystore.

The default implementation uses the Java Cryptographic Engine to provide access to the Java keystore. A custom implementation must implement the `com.sun.identity.saml.xmlsig.KeyProvider` interface.

Default value: `com.sun.identity.saml.xmlsig.JKSKeyProvider`

ssoadm attribute: `keyProviderClass`

3.1.3. Algorithms

The following settings appear on the **Algorithms** tab:

XML canonicalization algorithm

The algorithm used to canonicalize XML documents.

The possible values for this property are:

```
http://www.w3.org/2001/10/xml-exc-c14n#
http://www.w3.org/2001/10/xml-exc-c14n#WithComments
http://www.w3.org/TR/2001/REC-xml-c14n-20010315
http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
```

Default value: `http://www.w3.org/2001/10/xml-exc-c14n#`

ssoadm attribute: `canonicalizationAlgorithm`

XML signature algorithm

The algorithm used to sign XML documents.

The possible values for this property are:

```
http://www.w3.org/2000/09/xmldsig#rsa-sha1
http://www.w3.org/2000/09/xmldsig#hmac-sha1
http://www.w3.org/2000/09/xmldsig#dsa-sha1
http://www.w3.org/2001/04/xmldsig-more#rsa-md5
http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd160
http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
http://www.w3.org/2001/04/xmldsig-more#rsa-sha384
http://www.w3.org/2001/04/xmldsig-more#rsa-sha512
http://www.w3.org/2001/04/xmldsig-more#hmac-md5
http://www.w3.org/2001/04/xmldsig-more#hmac-ripemd160
http://www.w3.org/2001/04/xmldsig-more#hmac-sha256
http://www.w3.org/2001/04/xmldsig-more#hmac-sha384
http://www.w3.org/2001/04/xmldsig-more#hmac-sha512
```

Default value: `http://www.w3.org/2000/09/xmldsig#rsa-sha1`

ssoadm attribute: `signatureAlgorithm`

XML digest algorithm

The default digest algorithm to use in signing XML.

The possible values for this property are:

```
http://www.w3.org/2000/09/xmldsig#sha1
http://www.w3.org/2001/04/xmlenc#sha256
http://www.w3.org/2001/04/xmlenc#sha512
http://www.w3.org/2001/04/xmldsig-more#sha384
```

Default value: `http://www.w3.org/2000/09/xmldsig#sha1`

ssoadm attribute: `DigestAlgorithm`

Query String signature algorithm (RSA)

The default signature algorithm to use in case of RSA keys.

The possible values for this property are:

```
http://www.w3.org/2000/09/xmldsig#rsa-sha1
```

```
http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
http://www.w3.org/2001/04/xmldsig-more#rsa-sha384
http://www.w3.org/2001/04/xmldsig-more#rsa-sha512
```

Default value: `http://www.w3.org/2000/09/xmldsig#rsa-sha1`

ssoadm attribute: `QuerySignatureAlgorithmRSA`

Query String signature algorithm (DSA)

The default signature algorithm to use in case of DSA keys.

The possible values for this property are:

```
http://www.w3.org/2000/09/xmldsig#dsa-sha1
```

Default value: `http://www.w3.org/2000/09/xmldsig#dsa-sha1`

ssoadm attribute: `QuerySignatureAlgorithmDSA`

Query String signature algorithm (EC)

The default signature algorithm to use in case of EC keys.

The possible values for this property are:

```
http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1
http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256
http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384
http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512
```

Default value: `http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512`

ssoadm attribute: `QuerySignatureAlgorithmEC`

XML transformation algorithm

The algorithm used to transform XML documents.

The possible values for this property are:

```
http://www.w3.org/2001/10/xml-exc-c14n#
http://www.w3.org/2001/10/xml-exc-c14n#WithComments
http://www.w3.org/TR/2001/REC-xml-c14n-20010315
http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
http://www.w3.org/TR/1999/REC-xslt-19991116
http://www.w3.org/2000/09/xmldsig#base64
http://www.w3.org/TR/1999/REC-xpath-19991116
http://www.w3.org/2000/09/xmldsig#enveloped-signature
http://www.w3.org/TR/2001/WD-xptr-20010108
http://www.w3.org/2002/04/xmldsig-filter2
http://www.w3.org/2002/06/xmldsig-filter2
http://www.nue.et-inf.uni-siegen.de/~geuer-pollmann/#xpathFilter
```

Default value: `http://www.w3.org/2001/10/xml-exc-c14n#`

ssoadm attribute: `transformationAlgorithm`

3.1.4. Monitoring

The following settings appear on the **Monitoring** tab:

Monitoring Agent Provider Class

The Federation system uses this class to gain access to the monitoring system.

The default implementation uses the built-in OpenAM monitoring system. A custom implementation must implement the `com.sun.identity.plugin.monitoring.FedMonAgent` interface.

Default value: `com.sun.identity.plugin.monitoring.impl.AgentProvider`

ssoadm attribute: `monitoringAgentClass`

Monitoring Provider Class for SAML1

The SAMLv1 engine uses this class to gain access to the monitoring system

The default implementation uses the built-in OpenAM monitoring system. A custom implementation must implement the `com.sun.identity.plugin.monitoring.FedMonSAML1Svc` interface.

Default value: `com.sun.identity.plugin.monitoring.impl.FedMonSAML1SvcProvider`

ssoadm attribute: `monitoringSaml1Class`

Monitoring Provider Class for SAML2

The SAML2 engine uses this class to gain access to the monitoring system.

The default implementation uses the built-in OpenAM monitoring system. A custom implementation must implement the `com.sun.identity.plugin.monitoring.FedMonSAML2Svc` interface.

Default value: `com.sun.identity.plugin.monitoring.impl.FedMonSAML2SvcProvider`

ssoadm attribute: `monitoringSaml2Class`

Monitoring Provider Class for ID-FF

The ID-FF engine uses this class to gain access to the monitoring system.

The default implementation uses the built-in OpenAM monitoring system. A custom implementation must implement the `com.sun.identity.plugin.monitoring.FedMonIDFFSvc` interface.

Default value: `com.sun.identity.plugin.monitoring.impl.FedMonIDFFSvcProvider`

ssoadm attribute: `monitoringIdffClass`

Appendix A. Getting Support

For more information or resources about OpenAM and ForgeRock Support, see the following sections:

A.1. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.
- ForgeRock core documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

Core documentation therefore follows a three-phase review process designed to eliminate errors:

- Product managers and software architects review project documentation design with respect to the readers' software lifecycle needs.
- Subject matter experts review proposed documentation changes for technical accuracy and completeness with respect to the corresponding software.
- Quality experts validate implemented documentation changes for technical accuracy, completeness in scope, and usability for the readership.

The review process helps to ensure that documentation published for a ForgeRock release is technically accurate and complete.

Fully reviewed, published core documentation is available at http://backstage.forgerock.com/. Use this documentation when working with a ForgeRock Identity Platform release.

A.2. Joining the ForgeRock Community

Visit the Community resource center where you can find information about each project, download trial builds, browse the resource catalog, ask and answer questions on the forums, find community events near you, and find the source code for open source software.

A.3. Getting Support and Contacting ForgeRock

ForgeRock provides support services, professional services, classes through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see https://www.forgerock.com.

ForgeRock has staff members around the globe who support our international customers and partners. For details, visit https://www.forgerock.com, or send an email to ForgeRock at info@forgerock.com.

Glossary

Access control	Control to grant or to deny access to a resource.
Account lockout	The act of making an account temporarily or permanently inactive after successive authentication failures.
Actions	Defined as part of policies, these verbs indicate what authorized subjects can do to resources.
Advice	In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.
Agent administrator	User having privileges only to read and write policy agent profile configuration information, typically created to delegate policy agent profile creation to the user installing a policy agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	In general terms, a service exposing protected resources. In the context of OpenAM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.
Application type	Application types act as templates for creating policy applications. Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.

	Application types also define the internal normalization, indexing logic, and comparator logic for applications.
Attribute-based access control (ABAC)	Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.
Authentication	The act of confirming the identity of a principal.
Authentication chaining	A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.
Authentication level	Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.
Authentication module	OpenAM authentication unit that handles one way of obtaining and verifying credentials.
Authorization	The act of determining whether to grant or to deny a principal access to a resource.
Authorization Server	In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. OpenAM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. OpenAM can play this role in the OAuth 2.0 authorization framework.
Conditions	Defined as part of policies, these determine the circumstances under which which a policy applies. Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.

	Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.
Configuration datastore	LDAP directory service holding OpenAM configuration data.
Cross-domain single sign-on (CDSSO)	OpenAM capability allowing single sign-on across different DNS domains.
Delegation	Granting users administrative privileges with OpenAM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given subject in the context of a particular application, which actions are allowed and which are denied, and any related advice and attributes.
Extended metadata	Federation configuration information specific to OpenAM.
Extensible Access Control Markup Language (XACML)	Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.
Federation	Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and allowing principals to access services across different providers without authenticating repeatedly.
Fedlet	Service provider application capable of participating in a circle of trust and allowing federation without installing all of OpenAM on the service provider side; OpenAM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where OpenAM runs.
Identity	Set of data that uniquely describes a person or a thing such as a device or an application.
Identity federation	Linking of a principal's identity across multiple providers.
Identity provider (IdP)	Entity that produces assertions about a principal (such as how and when a principal authenticated, or that the principal's profile has a specified attribute value).
Identity repository	Data store holding user profiles and group information; different identity repositories can be defined for different realms.
Java EE policy agent	Java web application installed in a web container that acts as a policy agent, filtering requests to other applications in the container with policies based on application resource URLs.

Metadata	Federation configuration information for a provider.
Policy	Set of rules that define who is granted access to a protected resource when, how, and under what conditions.
Policy Agent	Agent that intercepts requests for resources, directs principals to OpenAM for authentication, and enforces policy decisions from OpenAM.
Policy Administration Point (PAP)	Entity that manages and stores policy definitions.
Policy Decision Point (PDP)	Entity that evaluates access rights and then issues authorization decisions.
Policy Enforcement Point (PEP)	Entity that intercepts a request for a resource and then enforces policy decisions from a PDP.
Policy Information Point (PIP)	Entity that provides extra information, such as user profile attributes that a PDP needs in order to make a decision.
Principal	Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities. When a Subject successfully authenticates, OpenAM associates the Subject with the Principal.
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified subjects in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	OpenAM unit for organizing configuration and identity information. Realms can be used for example when different parts of an organization have different applications and user data stores, and when different organizations use the same OpenAM deployment. Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.
Resource	Something a user can access over the network such as a web page. Defined as part of policies, these can include wildcards in order to match multiple actual resources.
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.

Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow OpenAM to return additional information in the form of "attributes" with the response to a policy decision.
Role based access control (RBAC)	Access control that is based on whether a user has been granted a set of permissions (a role).
Security Assertion Markup Language (SAML)	Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.
Service provider (SP)	Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).
Session	The interval that starts with the user authenticating through OpenAM and ends when the user logs out, or when their session is terminated. For browser-based clients, OpenAM manages user sessions across one or more applications by setting a session cookie. See also Stateful session and Stateless session.
Session high availability	Capability that lets any OpenAM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.
Session token	Unique identifier issued by OpenAM after successful authentication. For a Stateful session, the session token is used to track a principal's session.
Single log out (SLO)	Capability allowing a principal to end a session once, thereby ending her session across multiple applications.
Single sign-on (SSO)	Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.
Site	Group of OpenAM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability. Use sticky load balancing based on `amlbcookie` values to improve site performance. The load balancer can also be used to protect OpenAM services.
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateful session	An OpenAM session that resides in the Core Token Service's token store. Stateful sessions might also be cached in memory on one or

	more OpenAM servers. OpenAM tracks stateful sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.
Stateless session	An OpenAM session for which state information is encoded in OpenAM and stored on the client. The information from the session is not retained in the CTS token store. For browser-based clients, OpenAM sets a cookie in the browser that contains the session information.
Subject	Entity that requests access to a resource When a subject successfully authenticates, OpenAM associates the subject with the Principal that distinguishes it from other subjects. A subject can be associated with multiple principals.
User data store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service, a relational database, or a custom `IdRepo` implementation.
Web policy agent	Native library installed in a web server that acts as a policy agent with policies based on web page URLs.