Sun Java System SAML v2 Plug-in for Federation Services User's Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–5209–12
Jan. 2006

071218@19028

Contents

Preface

The *Sun™ Java System SAML v2 Plug-in for Federation Services User's Guide* provides information and procedures concerning this initial release of the Sun Java System SAML v2 Plug-in for Federation Services. The *User's Guide* includes installation instructions, administrative procedures, developer material, technical notes, and use cases.

Who Should Use This Book

The *Sun Java System SAML v2 Plug-in for Federation Services User's Guide* is intended for use by IT professionals, network administrators and software developers who wish to implement Security Assertion Markup Language (SAML) version 2 features in their deployment of Sun Java System Access Manager or Sun Java System Federation Manager server software. It is recommended that administrators understand the following technologies:

- Security Assertion Markup Language (SAML)
- Liberty Alliance Project Specifications for Federation and Web Services
- Java™ 2 Platform, Enterprise Edition (J2EE™ platform) web containers and corresponding deployment tools
- JavaServer Pages™
- HyperText Transfer Protocol (HTTP)
- HyperText Markup Language (HTML)
- eXtensible Markup Language (XML)
- Web Services Description Language (WSDL)
- SOAP

In addition, administrators should have a clear knowledge of the legal aspects of identity federation (including privacy laws) and corporate policies.

Before You Read This Book

The Sun Java System SAML v2 Plug-in for Federation Services can be installed on working instances of Sun Java System Access Manager and Sun Java System Federation Manager. Access Manager is a component of the Sun Java Enterprise System suite of products.

- If installing the SAML v2 Plug-in for Federation Services on an instance of Federation Manager, you should be familiar with the Sun Java System Federation Manager documentation.
- If installing the SAML v2 Plug-in for Federation Services on an instance of Access Manager, you should be familiar with the Sun Java System Access Manager documentation. Specifically, the *Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide* might be helpful.
- Because the Sun Java System SAML v2 Plug-in for Federation Services contains features based on the SAML specifications, you should be familiar with the SAML specifications.
- Because the Sun Java System SAML v2 Plug-in for Federation Services contains features based on the Liberty Alliance Project specifications, you should be familiar with the Liberty Alliance Project specifications.
- Because Access Manager is part of the Sun Java Enterprise System suite, you should be familiar with the Sun Java Enterprise System documentation.

How This Book Is Organized

The *Sun Java System SAML v2 Plug-in for Federation Services User's Guide* contains instructional and conceptual material regarding the use of the Sun Java System SAML v2 Plug-in for Federation Services. The book is organized into the chapters described in the following table.

TABLE P–1 Chapters in the SAML v2 Plug-in for Federation Services User's Guide

Chapter	Description
Chapter 1, "Introducing the SAML v2 Plug-in for Federation Services,"	An overview of the features of SAML v2 and the Sun Java System SAML v2 Plug-in for Federation Services.
Chapter 2, "Installing the SAML v2 Plug-in for Federation Services,"	Contains the installation procedures for the SAML v2 Plug-in for Federation Services, the SAML v2 IDP Discovery Service, and related information.
Chapter 3, "Administration,"	Contains information on how to set up and configure the SAML v2 Plug-in for Federation Services for SAML v2 communications.

TABLE P–1 Chapters in the SAML v2 Plug-in for Federation Services User's Guide *(Continued)*

Chapter	Description
Chapter 4, "Configuring Specialized Interactions,"	Contains deployment information and use cases for the SAML v2 Plug-in for Federation Services.
Chapter 5, "Developer Tools,"	Contains developer information for the SAML v2 Plug-in for Federation Services including how to install the software development kit, how to customize the JavaServer Pages, and how to implement the application programming interfaces and the service provider interfaces.
Appendix A, "Deploying the SAML v2 Plug-in for Federation Services Generated WAR"	Contains details on how to deploy a SAML v2 Plug-in for Federation Services generated WAR in the supported web containers.
Appendix B, "Log Message Reference"	Contains explanations of the messages written to the log files.

Related Books

The following titles are part of the documentation for the SAML v2 Plug-in for Federation Services. They can be found in the Sun Java System Access Manager 7 2005Q4 collection of manuals or the Sun Java System Federation Manager 7 2005Q4 collection of manuals on the Sun Microsystems documentation web site.

- The *Sun Java System SAML v2 Plug-in for Federation Services Release Notes* will be available online after the product is released. It gathers an assortment of last-minute information, including features and enhancements, known limitations and issues, and technical notes.
- The *Sun Java System SAML v2 Plug-in for Federation Services User's Guide* (this guide) is the core manual for the SAML v2 Plug-in for Federation Services. It includes instructions on how to install the SAML v2 Plug-in for Federation Services and information on how to use the features and manage the metadata. It also includes some basic deployment information in the form of use cases.
- The *Sun Java System SAML v2 Plug-in for Federation Services Java API Reference* are generated from Java code using the `JavaDoc` tool. These HTML pages provide information on the implementation of the Java packages.

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in the product documentation, contact Sun Support Services.

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Feedback

Sun Microsystems is interested in improving its documentation and welcomes your comments and suggestions. To share your thoughts, go to `http://docs.sun.com` and click the Send Comments link at the top of the page. In the online form provided, include the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the title of this book is *Sun Java System SAML v2 Plug-in for Federation Services User's Guide*, and the part number is 819–5209.

Accessing Sun Documentation Online

The docs.sun.com^{SM} web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

Ordering Sun Documentation

Sun Microsystems offers select product documentation in print. For a list of documents and how to order them, see “Buy printed documentation” at `http://docs.sun.com`.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–2 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–3 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`

TABLE P–3 Shell Prompts *(Continued)*

Shell	Prompt
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

Additional Sun Resources

For product downloads, professional services, patches, support, and additional developer information, go to the following locations:

- Download Center (`http://wwws.sun.com/software/download/`)
- Technical Support (`http://www.sun.com/service/support/software/`)
- Sun Java Systems Services Suite (`http://www.sun.com/service/sunjavasystem/sjsservicessuite.html`)
- Sun Enterprise Services, Solaris Patches, and Support (`http://sunsolve.sun.com/`)
- Developer Information (`http://developers.sun.com/prodtech/index.html`)

If you have technical questions about any Sun products, contact Sun Support and Services (`http://www.sun.com/service/contacting`).

CHAPTER 1

Introducing the SAML v2 Plug-in for Federation Services

Sun™ Java System SAML v2 Plug-in for Federation Services is an auxiliary program that works with either Sun Java System Access Manager or Sun Java System Federation Manager. The plug-in incorporates a subset of features based on the Security Assertion Markup Language (SAML) version 2 specifications and, when installed, allows support for interactions based on those specifications.

This chapter covers the following topics:

- "The SAML Standard" on page 13
- "Sun Java System SAML v2 Plug-in for Federation Services" on page 16
- "SAML v2 Plug-in for Federation Services Architecture" on page 19

The SAML Standard

SAML is an XML-based standard for communicating authentication, authorization and attribute information amongst online partners. It allows businesses to securely send data regarding the identity and entitlements of a principal between partnered organizations. The Organization for the Advancement of Structured Information Standards (OASIS) Security Services Technical Committee is in charge of defining, enhancing, and maintaining the specifications that define SAML. SAML v2 became an OASIS-approved standard in March, 2005, incorporating the definitions from SAML v1.x with functionality from other standards (including the Liberty ID-FF v1.2 and the Shibboleth initiative). Thus, SAML v2 also takes a critical step towards convergence of many of today's federated identity standards. The SAML v2 specifications can be found on the OASIS web site.

Basic SAML Components

SAML consists of a number of components that, when used together, permit the exchange of identity, authentication, and authorization information between autonomous organizations. The first component is an *assertion* which defines the structure and content of the information being transferred. The structure is based on the SAML v2 assertion schema. How an assertion is requested by, or pushed to, a service provider is defined as a request/response *protocol* encoded in its own structural guidelines: the SAML v2 protocol schema. A *binding* defines the communication protocols (such as HTTP or SOAP) over which the SAML protocol can be transported. Together, these three components create a *profile* (such as Web Browser Artifact or Web Browser POST). In general, profiles satisfy a particular use case. The following image illustrates how the components are integrated for a SAML interaction.

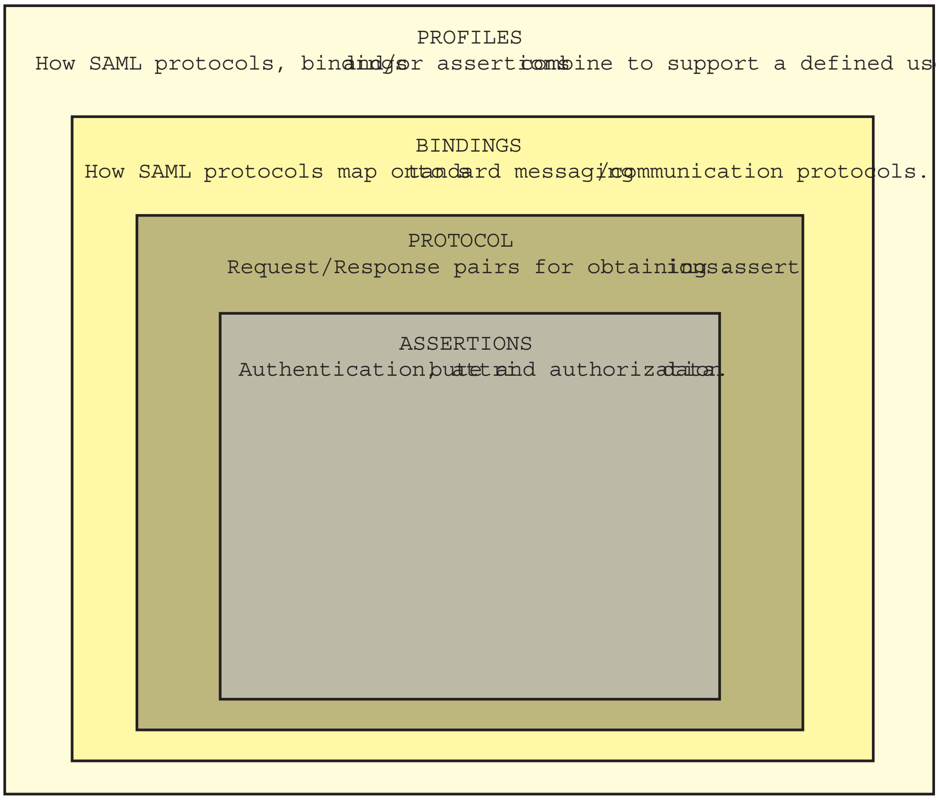
PROFILES
How SAML protocols, bindings and/or assertions combine to support a defined us
BINDINGS
How SAML protocols map onto standard messaging/communication protocols.
PROTOCOL
Request/Response pairs for obtaining assert
ASSERTIONS
Authentication, attribute and authorization data.

FIGURE 1–1 How Basic SAML Components Work Together

Two other components that may be included in SAML messages are:

Metadata

Metadata defines how configuration information shared between two communicating entities is structured. For instance, an entity's support for specific SAML bindings, identifier information, and public key information is defined in the metadata. The structure of the metadata is based on the SAML v2 metadata schema. The location of the metadata is defined by Domain Name Server (DNS) records.

Authentication Context

In some situations, one entity may want additional information to determine the authenticity of, and confidence in, the information being sent in an assertion. Authentication context permits the augmentation of assertions with information pertaining to the method of authentication used by the principal and how secure that method might be. For example, details of a multi-factor authentication can be included.

Note – More general information on SAML can be found at the OASIS web site or in the *Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide*.

SAML v2 Summary

The SAML v2 specifications are defined more broadly with particular attention paid to functionality dealing with federation. The new specifications clean up and make enhancements to existing functionality as well as integrating features previously defined in the Liberty ID-FF v1.2 and the Shibboleth initiative. Here is a summary of some of the concepts and features of SAML v2.

Note – This is a summary of features defined in the SAML v2 specifications. See "Key Features" on page 17 for a list of only those supported in the Sun Java System SAML v2 Plug-in for Federation Services product.

- Contains two concepts (appropriated from the Liberty ID-FF) that were not previously defined in the SAML specifications.
  - An *identity provider* is the system, or administrative domain, that asserts information about a subject. (It might also be referred to as a *SAML authority*, *authentication authority*, *attribute authority*, or *asserting party*). The identity provider might attest in a SAML assertion that a user has been authenticated because the user entered certain, associated attributes. For example, user John Doe has an email address of *john.doe@acompany.com*. The assertion from the identity provider might declare that John Doe was authenticated into the identity provider system by entering an email address and associated password.
  - A *service provider* is the system, or administrative domain, that relies on information supplied to it by an identity provider. (It might also be referred to as a *relying party*, or *assertion consumer*). It is the responsibility of the service provider to decide whether it trusts the identity provider from which SAML assertions are sent. The SAML assertion though does not dictate access; it is local policy that defines whether the subject may access local resources.
- Supports the following bindings for exchanging request/response pairs (protocol messages):
  - HTTP Redirect
  - HTTP POST

- SOAP
- HTTP Artifact

Caution – Previously, only an assertion could be exchanged using HTTP Artifact. Now, any protocol message can be exchanged using the artifact binding although small-sized messages and those that are not crucial (such as logout responses) should still not use it.

- Supports name identifier encryption (based on the Liberty ID-FF v1.x feature already developed for the Name ID Mapping protocol) and assertion encryption.
- Supports Reverse PAOS Binding in the Enhanced Client and Proxy (ECP) context.
- Supports unsolicited response.
- Allows use of pseudonyms (a key privacy-enabling technology) and name identifier management (for managing pseudonyms).

Note – Name identifier management encompasses both name registration and federation. These were separate features in the Liberty ID-FF v1.x.

- Supports an identity provider discovery profile for specifying an identity provider in deployments having more than one.
- Contains a metadata schema (for expressing configuration and trust-related data).
- Supports encryption of attribute statements, name identifiers, or entire assertions.
- Supports session management for single logout.
- Support for mobile standards.
- Supports affiliations.
- Supports the Transient Name Identifier Format
- Contains new attributes in the Name Identifier Policy, Identity Provider Proxying and Name Identifier Mapping protocols.

Sun Java System SAML v2 Plug-in for Federation Services

The Sun Java System SAML v2 Plug-in for Federation Services delivers a solution that allows businesses to establish a framework for sharing trusted information across a distributed network of partners using the standards-based SAML v2. Towards this end, HTTP(S)-based service endpoints and SOAP service endpoints are supplied in support of the respective profiles defined in the specifications as well as assertion and protocol object manipulating classes.

After installing the SAML v2 Plug-in for Federation Services, a group of companies can exchange security assertions for single sign-on when they all participate in the same SAML v2–enabled circle of trust. A web browser can access all HTTP(S)-based service endpoints and an application can make use of the SOAP endpoints and classes as long as metadata for each participating business on BOTH sides of the interaction is exchanged beforehand.

Key Features

The key features of the SAML v2 Plug-in for Federation Services include:

- Interoperability with the following Sun Microsystems' server products:
  - Access Manager 7 2005Q4
  - Federation Manager 7.0

  Note – The SAML v2 Plug-in for Federation Services supports all web containers and platforms used by these products.

- Single sign-on using the POST profile, the Artifact binding (also referred to as HTTP redirect), and unsolicited responses (initiated by the identity provider).
- Single logout using HTTP redirect and SOAP binding.
- Federation termination using HTTP redirect and SOAP binding.
- Auto-federation (automatic linking of service provider and identity provider user accounts based on a common attribute).
- Bulk federation.
- Supports one-time federation (transient NameID format in SSO).
- Service provider interfaces (SPI) for the following:
  - Account mapping (map between the account referred to in the incoming request and the local user account).
  - Attribute mapping (specifies which set of user attributes in an identity provider user account need to be included in an assertion AND maps the attributes included in an assertion by the identity provider to attributes in the user account defined by the service provider).
  - Authentication context mapping (map between Authentication Contexts defined in the SAML v2 specifications and authentication framework schemes defined in Access Manager and Federation Manager (user/module/service/role/level based authentication).
- Supports Basic Authentication, SSL and SSL with client authentication for SOAP Binding.
- SAML v2 authentication module.
- Support for the identity provider Discovery Protocol as the SAML v2 IDP Discovery Service.
- Supports SAML v2 Circle of Trust.

- A SAML v2 software development kit (SDK).
- XML verification, signing, encryption and decryption.
- JavaServer Pages™ (JSP™) for profile initiation and processing.
- Support for load balancing.
- Pre-deployment of sample.
- Protocol coexistence with the SAML v1.x and the Liberty Alliance Project's Identity Federation Framework (Liberty ID-FF).

Note – Although the SAML v2 and SAML v1.x specifications can coexist, they are not interoperable.

Product Requirements

You can install the SAML v2 Plug-in for Federation Services on the following products and platforms.

Sun Java System Access Manager

The SAML v2 Plug-in for Federation Services is supported on Sun Java System Access Manager 7 2005Q4 in both realm and legacy mode. It can be installed on versions of the Solaris™ Operating System (OS) or Red Hat™ Linux.

TABLE 1–1 Access Manager Requirements

Hardware	Operating System
SPARC®	Solaris™ Operating System (OS) 8 / 9 / 10
x86	▪ Solaris OS 9 / 10 ▪ Red Hat™ Linux, WS / AS / ES 2.1 Update 6 or later ▪ Red Hat Linux, WS / AS / ES 3.0 ▪ Red Hat Linux, WS / AS / ES 3.0 Update 1 / 2 / 3 / 4

Sun Java System Federation Manager

The SAML v2 Plug-in for Federation Services is supported on Sun Java System Federation Manager 7 2005Q4. It can be installed on versions of the Solaris OS.

TABLE 1–2 Federation Manager Requirements

Hardware	Operating System
SPARC	Solaris OS 8 / 9 / 10
x86	Solaris OS 9 / 10

SAML v2 Plug-in for Federation Services Architecture

The SAML v2 Plug-in for Federation Services consists of web-based services [using SOAP, XML over HTTP(S) or HTML over HTTP(S)], and Java™-based application provider interfaces (API) and service provider interfaces (SPI). The figure below illustrates this architecture. Additionally, the figure shows an agent embedded into a web container in which a service provider application is deployed. This agent enables the service provider to participate in the SAML or Liberty-based protocols.

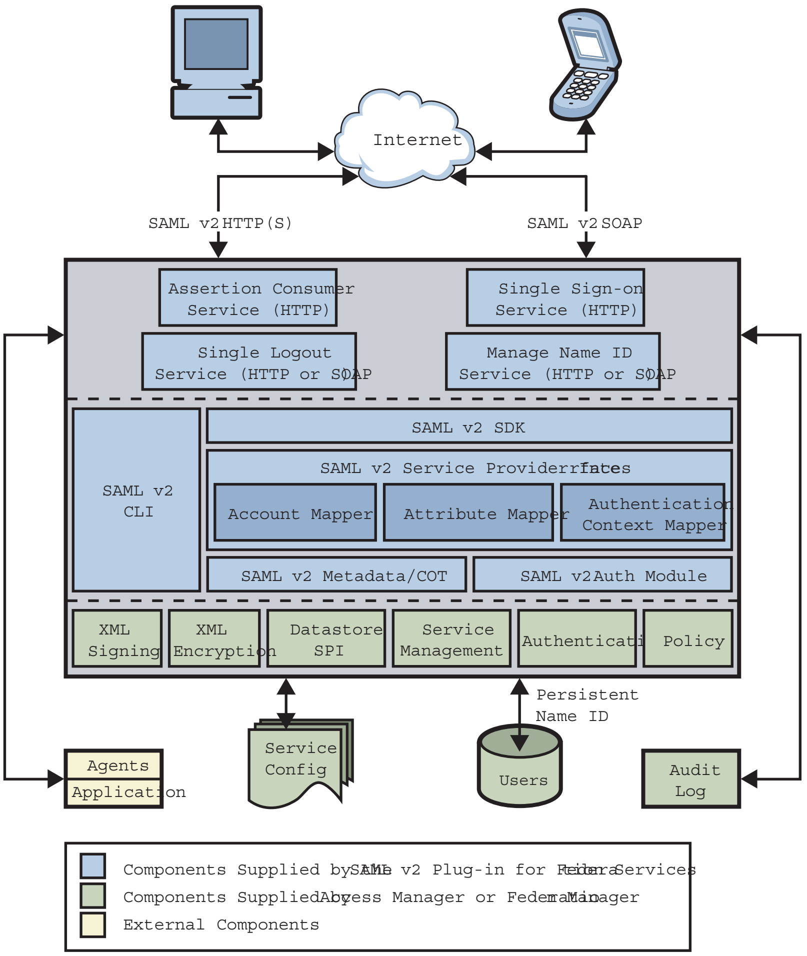
Internet
SAML v2 HTTP(S)
SAML v2 SOAP
Assertion Consumer Service (HTTP)
Single Sign-on Service (HTTP)
Single Logout Service (HTTP or SOAP)
Manage Name ID Service (HTTP or SOAP)
SAML v2 SDK
SAML v2 CLI
SAML v2 Service Provider Interfaces
Account Mapper
Attribute Mapper
Authentication Context Mapper
SAML v2 Metadata/COT
SAML v2 Auth Module
XML Signing
XML Encryption
Datastore SPI
Service Management
Authentication
Policy
Persistent Name ID
Agents
Application
Service Config
Users
Audit Log
Components Supplied by SAML v2 Plug-in for Federation Services
Components Supplied by Access Manager or Federation Manager
External Components

FIGURE 1–2 SAML v2 Plug-in for Federation Services Architecture

Installation

A single Solaris package will be supplied along with installation and configuration scripts to setup the SAML v2 protocol endpoints as well as the SAML v2 metadata for the identity provider and the service provider. The SAML v2 Plug-in for Federation Services will leverage the core infrastructure of the underlying server product (Access Manager or Federation Manager) for authentication, authorization, service management, logging/auditing, delegation, and access to user data stores. The SAML v2 Plug-in for Federation Services is implemented as a Solaris or Linux package. Scripts to uninstall the plug-in are also supplied. More information can be found in Chapter 2, "Installing the SAML v2 Plug-in for Federation Services."

Administration

In order to communicate using the SAML v2 profiles you need, at least, two instances of the installed SAML v2 Plug-in for Federation Services. One instance will act for the identity provider and the other will act for the service provider. To prepare your instances of the SAML v2 Plug-in for Federation Services for interactions, you need to exchange configuration information or *metadata* with all participating identity and service providers, import each provider's metadata using an XML-based *metadata configuration file*, and assemble the providers into a *circle of trust*. The SAML v2 Plug-in for Federation Services accomplishes all this administration and configuration using the command-line interface, `saml2meta`. Utility APIs can then be used to communicate with the data store, reading, writing, and managing the relevant properties and property values. More information can be found in Chapter 3, “Administration.”

Note – Membership in a circle of trust is transient and might change over the life cycle of the circle as relationships among the partners themselves change.

Java Developer Tools

The SAML v2 Plug-in for Federation Services includes Java programming tools for developer access to the SAML v2 features. They include:

- “Interfaces” on page 21
- “JavaServer Pages” on page 22

Interfaces

The following sections contain general information about the interfaces provided with the SAML v2 Plug-in for Federation Services.

- “Application Programming Interfaces” on page 21
- “Service Provider Interfaces” on page 22

More information can be found in Chapter 5, “Developer Tools,” and the *Sun Java System SAMLv2 Plug-in for Federation Services Java API Reference*.

Application Programming Interfaces

The SAML v2 Plug-in for Federation Services provides a software development kit (SDK) containing API that can be used to construct and process assertions, requests, and responses. The SDK is designed to be plugged in although it can also be installed and run as a standalone application (without an instance of Access Manager or Federation Manager). Here is a list of the included Java API packages:

- `com.sun.identity.saml2.assertion`

- `com.sun.identity.saml2.common`
- `com.sun.identity.saml2.protocol`

More information about the SAML v2 Plug-in for Federation Services SDK can be found in "The SAML v2 Plug-in for Federation Services SDK" on page 93 and the *Sun Java System SAMLv2 Plug-in for Federation Services Java API Reference*.

Service Provider Interfaces

The SAML v2 Plug-in for Federation Services provides SPI that can be implemented for application development. They are collected in the package `com.sun.identity.saml2.plugins`. Default implementations are provided out-of-the-box although customized implementations can be developed by modifying the appropriate attribute in the extended metadata configuration file. More information about the SPI can be found in "Service Provider Interfaces" on page 97 and the *Sun Java System SAMLv2 Plug-in for Federation Services Java API Reference*.

JavaServer Pages

The SAML v2 Plug-in for Federation Services provides JSP that can be used to initiate single sign-on, single logout and termination requests from either the identity provider or the service provider using a web browser. The JSP accept query parameters to allow flexibility in constructing the requests and can be modified for your deployment. More information about the JSP can be found in "JavaServer Pages" on page 103.

CHAPTER 2

Installing the SAML v2 Plug-in for Federation Services

The Sun Java System SAML v2 Plug-in for Federation Services installs into an instance of Sun Java System Access Manager 7 2005Q4 or Sun Java System Federation Manager 7. Once deployed, it extends the functionality of the server product to include SAML v2–based interactions.

This chapter contains the following topics:

Installation Process

When the SAML v2 Plug-in for Federation Services is installed, the SAML v2 package is integrated into either the existing Access Manager or Federation Manager web archive (WAR). This modified WAR is then redeployed to add SAML v2 functionality to the existing application. Following is an overview of the process to install the SAML v2 Plug-in for Federation Services.

1. Install Sun Java System Access Manager 7 2005Q4 or Sun Java System Federation Manager 7 2005Q4 into one of the supported web containers.

   For more information, see "Supported Server Products" on page 24 and "Supported Web Containers" on page 27.
2. Download and unpack the SAML v2 Plug-in for Federation Services bits.

See the *Sun Java System SAML v2 Plug-in for Federation Services Release Notes* for the download URL.

3. Create an installation configuration properties file based on the included `saml2silent` template.

   For more information, see "Creating an Installation Configuration Properties File" on page 28.

4. Run `saml2setup` to install the plug-in packages and create a new Access Manager or Federation Manager WAR.

   For more information, see "Installing the SAML v2 Plug-in for Federation Services" on page 30.

5. Deploy the modified WAR to your web container and restart it.

   For more information, see Appendix A, "Deploying the SAML v2 Plug-in for Federation Services Generated WAR".

6. Make any postinstallation modifications to your instance of Access Manager or Federation Manager.

   For more information, see "Postinstallation" on page 33.

After completing the installation, see Chapter 3, "Administration," and Chapter 4, "Configuring Specialized Interactions," for instructions on how to use the SAML v2 Plug-in for Federation Services for SAML v2 interactions.

Supported Software Products

This section covers information you should know about the supported software products. It includes the following topics:

- "Supported Server Products" on page 24
- "Supported Web Containers" on page 27

Supported Server Products

The SAML v2 Plug-in for Federation Services can only be installed in a running instance of either Sun Java System Access Manager 7 2005Q4 or Sun Java System Federation Manager 7 2005Q4. These products can be installed on versions of the Solaris™ Operating System (OS), Red Hat™ Linux, or Microsoft Windows. The following sections contain information regarding installation of the SAML v2 Plug-in for Federation Services on these server products.

- "Installing on Sun Java System Access Manager 7 2005Q4" on page 25
- "Installing on Sun Java System Federation Manager 7 2005Q4" on page 26

Installing on Sun Java System Access Manager 7 2005Q4

This section contains information on an Access Manager installation and preparing it for the SAML v2 Plug-in for Federation Services installation.

- "Installing Access Manager" on page 25
- "Preparing Access Manager for SAML v2 Plug-in for Federation Services" on page 25

Note – The SAML v2 Plug-in for Federation Services works in either the realm or legacy mode of Access Manager.

Installing Access Manager

Access Manager can be installed on versions of the Solaris™ Operating System (OS), Red Hat™ Linux, or Microsoft Windows. The default installation directories for Solaris and Linux are `/opt/SUNWam/` and `/opt/sun/identity/`, respectively. `/opt` is the root installation directory. `SUNWam` and `sun/identity` are the product directories.

Note – `/opt` is the default root installation directory for Access Manager. When specified in a path, `/`*AccessManager-base* will be used as a generic placeholder. `/SUNWam` and `/sun/identity` are the default product directories for Access Manager. When specified in a path, `/`*product-directory*`/` will be used as a generic placeholder. See the *Sun Java System Access Manager 7 2005Q4 Deployment Planning Guide* for more information on the product's installed layout.

For information on installing Access Manager, see the *Sun Java Enterprise System 2005Q4 Installation Guide for UNIX*.

Preparing Access Manager for SAML v2 Plug-in for Federation Services

Before installing the SAML v2 Plug-in for Federation Services into an instance of Sun Java System Access Manager, you need to copy the `amserver.war` (located in `/`*AccessManager-base*`/`*product-directory*`/`) into a new directory and extract its contents using the following command:

```
jar –xvf amserver.war
```

This new directory is referred to as the *staging directory* and contains the original contents for your Access Manager application. Any modifications to the application must be made to the files in this directory. These files are then repackaged into a new WAR and redeployed into your web container. The path to this directory is the value used for the `STAGING_DIR` variable in your installation file.

Caution – If your instance of Access Manager has been customized, be sure to copy and extract the updated WAR.

For more information on Linux and Windows installations of the SAML v2 Plug-in for Federation Services, see the following:

- *Technical Note: Sun Java System SAML v2 Plug-in for Federation Services on Linux*
- *Technical Note: Sun Java System SAML v2 Plug-in for Federation Services on Windows*

Installing on Sun Java System Federation Manager 7 2005Q4

This section contains information on a Federation Manager installation and preparing it for the SAML v2 Plug-in for Federation Services installation.

- "Installing Federation Manager" on page 26
- "Preparing Federation Manager for SAML v2 Plug-in for Federation Services" on page 26

Installing Federation Manager

Federation Manager can be installed on the Solaris OS, Red Hat Linux, or Microsoft Windows. The default root installation directory for the Solaris OS is `/opt`.

Note – `/opt` is the default root installation directory for Federation Manager. When specified in a path, /*FederationManager-base* will be used as a generic placeholder. Unlike in Access Manager, the `SUNWam` directory in Federation Manager cannot be modified so references to this directory will remain static. See the *Sun Java System Federation Manager 7.0 User's Guide* for more information on the default installation directory and the default runtime directory.

For information on installing Federation Manager, see the *Sun Java System Federation Manager 7.0 User's Guide*. Additional information on Linux and Windows installations of Federation Manager can be found in the following:

- *Technical Note: Sun Java System Federation Manager 7.0 on Linux*
- *Technical Note: Sun Java System Federation Manager 7.0 on Windows*

Preparing Federation Manager for SAML v2 Plug-in for Federation Services

Before installing the SAML v2 Plug-in for Federation Services into an instance of Sun Java System Federation Manager, backup your staging directory. For more information on Linux and Windows installations of the SAML v2 Plug-in for Federation Services, see the following:

- *Technical Note: Sun Java System SAML v2 Plug-in for Federation Services on Linux*
- *Technical Note: Sun Java System SAML v2 Plug-in for Federation Services on Windows*

Supported Web Containers

A WAR modified with the SAML v2 Plug-in for Federation Services can be deployed in any of the following web containers. CPU and memory requirements are based on the needs of the web container. See your product's documentation for installation information.

TABLE 2–1 Supported Web Containers

Web Container	Minimum Version
Sun Java System Web Server	6.1sp4
Sun Java System Application Server	8.1
BEA WebLogic® Server	8.1
WebSphere® Application Server	5.1

The `saml2setup` Command-line Reference

`saml2setup` is the command-line installer for the SAML v2 Plug-in for Federation Services. By default, it:

- Installs the SAML v2 Plug-in for Federation Services packages on an instance of supported versions of Access Manager or Federation Manager.
- Configures the plug-in and generates a modified WAR for redeployment.

The `saml2setup` syntax is:

`saml2setup install [-p]  -s` *installation-configuration-properties-file*

`saml2setup uninstall -s` *installation-configuration-properties-file*

`saml2setup configure -s` *installation-configuration-properties-file*

`saml2setup unconfigure -s` *installation-configuration-properties-file*

`saml2setup -V`

`saml2setup -?`

Note – `saml2setup` takes as input an installation configuration properties file. See "Creating an Installation Configuration Properties File" on page 28 for more information.

`-p`	Specifies that the SAML v2 Plug-in for Federation Services packages will be installed only. The plug-in will not be configured and the modified WAR will not be generated.
`-s` or `--silent`	Specifies the name of the installation configuration properties file used as input to the installer. See "Creating an Installation Configuration Properties File" on page 28
`-V`	Displays version information.
`-?`	Displays help information.

The subcommands of `saml2setup` are:

`install`	Installs the SAML v2 Plug-in for Federation Services packages, configures the plug-in, and generates a WAR. **Note –** The function of this subcommand will change when run using the `-p` option.
`uninstall`	Removes the SAML v2 Plug-in for Federation Services and its packages.
`configure`	Configures the plug-in and generates a WAR only.
`unconfigure`	Removes a configured SAML v2 Plug-in for Federation Services and WAR but leaves the packages, allowing for future reconfiguration using the `configure` subcommand.

Creating an Installation Configuration Properties File

To install the SAML v2 Plug-in for Federation Services you must have an installation configuration properties file based on the template `saml2silent`. The `saml2silent` template is included with the installation binaries and can be modified based on your deployment. The `saml2silent` template can be found on the top-level of the directory to which the binaries were unpacked. Following is a sample installation configuration properties file that might be used to install the SAML v2 Plug-in for Federation Services on an instance of Federation Manager. Descriptions of the properties themselves can be found in Table 2–2.

```
############### START OF VARIABLE DEFINITIONS ###############

STAGING_DIR=/var/opt/SUNWam/fm/war_staging
ADMINPASSWD=11111111
DEPLOY_SAMPLES=true
SYSTEM=FM
AM_INSTANCE=
LOAD_SCHEMA=true
DS_DIRMGRDN="cn=Directory Manager"
```

```
DS_DIRMGRPASSWD=22222222
IDPDISCOVERY_ONLY=false
COMMON_COOKIE_DOMAIN=
COOKIE_ENCODE=true

############### END OF VARIABLE DEFINITIONS ###############
```

Note – Your modified installation file is used as input to the installer utility, `saml2setup`. More information on the installer utility can be found in "Installing the SAML v2 Plug-in for Federation Services" on page 30.

TABLE 2–2 `saml2silent` Installation File Property Definitions

Property	Definition
`STAGING_DIR`	Defines the staging directory for the SAML v2 Plug-in for Federation Services WAR. ■ The default staging directory on Access Manager is `/var/opt/`*product-directory*`/fm/war_staging`. For information specific to this variable when installing on Access Manager, see "Installing on Sun Java System Access Manager 7 2005Q4" on page 25. ■ The default staging directory on Federation Manager is `/var/opt/SUNWam/fm/war_staging`. For information specific to this variable when installing on Federation Manager, see "Installing on Sun Java System Federation Manager 7 2005Q4" on page 26.
`ADMINPASSWD`	Specifies the password chosen for the underlying product's administrator; by default, `amadmin`.
`DEPLOY_SAMPLES`	Defines whether the included sample will be deployed as part of the installation. The default value is `true`.
`SYSTEM`	Defines the server product into which the plug-in will be installed. It takes a value of `AM` if installing into an instance of Access Manager or `FM` if installing into an instance of Federation Manager. If no value is specified, the installer will automatically detect the server product.
`AM_INSTANCE`	Used if there are multiple instances of Access Manager. The value would be the name of the particular instance. If no value is specified, the installer will automatically detect the first instance of Access Manager. **Note –** This variable has no relevance when installing into an instance of Federation Manager.
`LOAD_SCHEMA`	Defines whether or not to automatically load the LDAP schema. The default value is `true`. There are instances when you might load the LDAP schema manually. For example, if `ldapmodify` is not available, you might set `LOAD_SCHEMA` to `false`.

TABLE 2–2 `saml2silent` Installation File Property Definitions *(Continued)*

Property	Definition
`DS_DIRMGRDN`	Defines the distinguished name (DN) of the user that has permissions to bind to the LDAP directory. This is required when `LOAD_SCHEMA` is `true`.
`DS_DIRMGRPASSWD`	Defines the password associated with the user DN that will bind to the LDAP directory. This is required when `LOAD_SCHEMA` is `true`. **Caution** – The value of this property is very sensitive. Be sure to protect the password after installation by removing it entirely from the file, or protect the file itself by setting the appropriate permissions.
`IDPDISCOVERY_ONLY`	Defines whether the installer will configure the SAML v2 Plug-in for Federation Services or the SAML v2 IDP Discovery Service only. If `true`, only the SAML v2 IDP Discovery Service will be configured. If `false`, the full SAML v2 Plug-in for Federation Services will be configured. The default value is `false`. **Note** – For more information on the SAML v2 IDP Discovery Service, see "Installing the SAML v2 IDP Discovery Service" on page 32 and "The SAML v2 IDP Discovery Service" on page 55.
`COMMON_COOKIE_DOMAIN`	Defines the common domain for the SAML v2 IDP Discovery Service. The value of this property must be set to `.`*cookie-domain-name* as in `.sun.com`.
`COOKIE_ENCODE`	Defines whether the common domain cookie will be URL encoded before setting and URL decoded before reading. If set to `true` the SAML v2 IDP Discovery Service will encode the cookie before setting and decode it before reading. If set to `false`, the SAML v2 IDP Discovery Service will not encode or decode the cookie. It will be set and received as is.

Installing the SAML v2 Plug-in for Federation Services

To install the SAML v2 Plug-in for Federation Services packages and create an updated WAR, run the `saml2setup` installer as described in "The `saml2setup` Command-line Reference" on page 27.

Note – `saml2setup` takes as input an installation configuration properties file. See "Creating an Installation Configuration Properties File" on page 28 for more information.

▼ To Install the SAML v2 Plug-in for Federation Services

Before You Begin Create an installation configuration properties file as described in "Creating an Installation Configuration Properties File" on page 28.

Note – For instances of Access Manager and Federation Manager using an LDAPv3–compliant directory, the new schema file is loaded only if the `LOAD_SCHEMA` variable in the installation configuration properties file is set to `true`.

---

1 Log in as root.

You must have system administrator privileges to run the SAML v2 Plug-in for Federation Services installer.

2 Create a new directory.

```
# mkdir saml2bits

# cd saml2bits
```

3 Download the *file-name*`.tar.gz` file into the new directory.

See the *Sun Java System SAML v2 Plug-in for Federation Services Release Notes* for the download URL.

4 Unpack the product binaries by typing:

```
# gunzip –dc file-name.tar.gz | tar -xvof -
```

where *file-name*`.tar.gz` is the name of the downloaded file.

5 Run the `saml2setup` installer as follows:

```
# saml2setup install -s installation-file-name
```

where *installation-file-name* is the name of the installation configuration properties file described in "Creating an Installation Configuration Properties File" on page 28.

The installer will install the packages, configure the plug-in, and create an updated WAR using the service deployment identifier specified in the `AMConfig.properties` file of the specific server product.

- When installed into an instance of Access Manager, the new WAR is located in `/`*AccessManager-base*`/`*product-directory*`/` and is called *service-deploy-uri*`.war` as in, for example, `amserver.war`.

Note – `AMConfig.properties` is located in the `/etc/opt/`*product-directory*`/config` directory.

- When installed into an instance of Federation Manager, the new WAR is located in the staging directory defined in the installation configuration properties file and is called *FM-deploy-uri*`.war` as in, for example, `federation.war`

Note – `AMConfig.properties` is located in the /*staging-directory*/`web-src/WEB-INF/classes` directory.

---

6 **Follow the instructions in Appendix A, "Deploying the SAML v2 Plug-in for Federation Services Generated WAR" to deploy the modified WAR and complete the installation.**

Next Steps Restart Federation Manager if you installed the SAML v2 Plug-in for Federation Services on that server product.

Installing the SAML v2 IDP Discovery Service

In deployments having more than one identity provider, the SAML v2 IDP Discovery Service allows service providers to determine which identity provider a principal uses with the Web Browser SSO profile. The SAML v2 IDP Discovery Service relies on a cookie written in a domain common to all identity providers and service providers in a circle of trust. The SAML v2 Plug-in for Federation Services can install a standalone instance of the SAML v2 IDP Discovery Service in this predetermined domain, also known as the *common domain*.

Note – Information on configuring SAML v2 IDP Discovery Service is in "The SAML v2 IDP Discovery Service" on page 55.

▼ To Install the SAML v2 IDP Discovery Service

Before You Begin Download and unpack the SAML v2 Plug-in for Federation Services binaries as described in "Installing the SAML v2 Plug-in for Federation Services" on page 30.

1 **Create an installation configuration properties file.**

Be sure to set the `IDPDISCOVERY_ONLY`, `COMMON_COOKIE_DOMAIN`, and `COOKIE_ENCODE` properties as described in "Creating an Installation Configuration Properties File" on page 28.

2 **Run the `saml2setup` command.**

```
# saml2setup install -s installation-file-name
```

where *installation-file-name* is the name of the installation configuration properties file described in "Creating an Installation Configuration Properties File" on page 28.

The installer will create a SAML v2 IDP Discovery Service WAR named `idpdiscovery.war` in /*AccessManager-base*/*product-directory*/`saml2/` or /*FederationManager-base*/`SUNWam/saml2/`.

3 **Deploy `idpdiscovery.war` according to the instructions in Appendix A, "Deploying the SAML v2 Plug-in for Federation Services Generated WAR".**

4 **Restart your web container.**

Next Steps See "The SAML v2 IDP Discovery Service" on page 55 to configure the SAML v2 IDP Discovery Service.

Postinstallation

This section contains postinstallation tasks specific to the product on which you are installing the SAML v2 Plug-in for Federation Services. It includes the following topics:

- "Access Manager Postinstallation" on page 33
- "Federation Manager Postinstallation" on page 36

After completing these tasks, see Chapter 3, "Administration," to begin configuring your instance of the SAML v2 Plug-in for Federation Services.

Access Manager Postinstallation

The following sections contain some procedures to perform after installing Access Manager.

- "Adding the `sunFMSAML2NameIdentifier` Object Class" on page 33
- "Enabling the SAML v2 Authentication Module" on page 34

Adding the `sunFMSAML2NameIdentifier` Object Class

If installing the SAML v2 Plug-in for Federation Services on an instance of Access Manager that uses an LDAPv3-compliant directory for a user data store, you must add the `sunFMSAML2NameIdentifier` object class to all existing users. This object class contains two attributes:

- `sun-fm-saml2—nameid-info-key` is used for searching purposes. The attribute's value takes the form *hosted-entity-id*|*remote-entity-id*|*idp-nameid*.
- `sun-fm-saml2—nameid-info` is used to store all information related to the name identifier. The attribute's value takes the form *hosted-entity-id*|*remote-entity-id*|*idp-nameid*|*idp-nameid-qualifier*|*idp-nameid-format*|*sp-nameid*|*sp-nameid-qualifier*|*hosted-entity-role*|*is-affiliation*.

Note – The values in these attributes are defined in the SAML v2 specifications. For example, *hosted-entity-role* takes a value of `IDPRole` or `SPRole` (based on the configuration of the provider) and *is-affiliation* specifies whether the federation is affiliation-based (taking a value of `true` or `false`). For explanations on the other attributes, see the Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 specification.

To add `sunFMSAML2NameIdentifier` to the default `amadmin` entry, you would run `ldapmodify` (available in the `bin` directory) using the following LDIF as input:

```
DN: uid=amadmin,ou=people,dc=sun,dc=com
changetype: modify
add: objectclass
objectclass: sunFMSAML2NameIdentifier
```

This task is not required for installations of Access Manager 7.1. It is also not required for installations of Federation Manager that use an LDAPv3-compliant directory as a user data store because the object class is automatically added if not found.

Caution – Be sure to set your class path correctly before using `ldapmodify`.

Enabling the SAML v2 Authentication Module

If installing the SAML v2 Plug-in for Federation Services on an instance of Access Manager, you might have to enable the SAML v2 authentication module using the Access Manager console. The following sections explain the procedure to do this in both legacy and realm mode.

- “To Enable the SAML v2 Authentication Module in Legacy Mode” on page 34
- “To Enable the SAML v2 Authentication Module in Legacy Mode Using `amadmin`” on page 35
- “To Enable the SAML v2 Authentication Module in Realm Mode” on page 35

Note – This is only necessary for instances of Access Manager acting as service providers.

▼ To Enable the SAML v2 Authentication Module in Legacy Mode

1. **Log in to Access Manager as the top-level administrator, by default,** `amadmin`**.**

2. **Select the Identity Management tab.**

3. **Select Services from the View drop down box.**

4 **Click Add Service.**

5 **Select SAML2 and click OK.**

6 **Log out of the Access Manager console.**

▼ To Enable the SAML v2 Authentication Module in Legacy Mode Using `amadmin`

You can also enable the SAML v2 authentication module using the `amadmin` command line tool.

1 **Save the following XML code to a file.**

```
<Requests>
<OrganizationRequests DN="<root_suffix>">
   <RegisterServices>
       <Service_Name>sunAMAuthSAML2Service</Service_Name>
   </RegisterServices>
</OrganizationRequests>
</Requests>
```

2 **Load the XML file using the `amadmin` command line tool to register the SAMLv2 authentication module.**

For information on the `amadmin` command line tool, see Chapter 1, "The amadmin Command Line Tool," in *Sun Java System Access Manager 7.1 Administration Reference*.

▼ To Enable the SAML v2 Authentication Module in Realm Mode

The SAML v2 Authentication Module is enabled during installation of Access Manager in Realm mode. The following procedure is for informational purposes.

1 **Log in to Access Manager as the top-level administrator, by default, `amadmin`.**

2 **Select the Access Control tab.**

3 **Select the appropriate realm.**

4 **Click the Authentication tab.**

5 **Click New under Module Instances.**

6 **Enter SAML2 as a name.**

7 **Select the SAML2 radio button.**

8 **Click Create.**

9 **Click Save.**

10 **Log out of the Access Manager console.**

Federation Manager Postinstallation

If installing the SAML v2 Plug-in for Federation Services on an instance of Federation Manager, you must enable the SAML v2 authentication module using the Federation Manager console.

▼ To Enable the SAML v2 Authentication Module

1 **Log in to Federation Manager as the top-level administrator, by default,** `amadmin`**.**

2 **Select the Organization tab.**

3 **Select the Authentication tab.**

4 **Click Edit next to the Core service.**

5 **Select SAML2 in the Organization Authentication Modules attribute list and click Save.**

6 **Log out of the Federation Manager console.**

Uninstalling the SAML v2 Plug-in for Federation Services

To uninstall the SAML v2 Plug-in for Federation Services, run the following command using the same installation configuration properties file for input that was used as input during installation.

```
# saml2setup uninstall -s installation-file-name
```

This command removes the SAML v2 Plug-in for Federation Services and its packages.

CHAPTER 3

Administration

After installing the SAML v2 Plug-in for Federation Services, follow the procedures in this chapter to configure the product for SAML v2 interactions. It covers the following topics:

Provider Metadata and Circles of Trust

In order to communicate using the SAML v2 profiles you need, at the least, two instances of the installed SAML v2 Plug-in for Federation Services configured in one circle of trust. Circles of trust configured for real time interactions must have, at the least, one instance acting as the circle's identity provider and one instance acting as a service provider.

To prepare your instances of the SAML v2 Plug-in for Federation Services for interactions, you need to exchange and import the metadata for all participating identity and service providers, and assemble the providers into a circle of trust. The following steps are an overview of the process. Many of these steps are accomplished using the `saml2meta` interface as described in "The `saml2meta` Command-line Reference" on page 57.

Note – Before beginning this configuration process, be sure that you have completed the instructions in "Postinstallation" on page 33.

1. Decide whether the instance of the SAML v2 Plug-in for Federation Services you are configuring will act as either an identity provider, a service provider, or both.
2. Create standard and extended metadata configuration files containing the appropriate metadata for your organization as described in "Metadata" on page 38.

Tip – Alternately, you can use the default identity provider metadata configuration files (`idpMeta.xml` and `idpExtended.xml`) or the default service provider metadata configuration files (`spMeta.xml` and `spExtended.xml`) created during installation. See "Standard Metadata Properties" on page 39 and "Extended Metadata Properties" on page 41 for more information.

---

3. Create a circle of trust as described in "Managing Circles of Trust using `saml2meta`" on page 60.

   Information about circles of trust can be found in "Circles of Trust" on page 54.
4. Import your organization's provider metadata into the circle of trust as described in "Managing Metadata using `saml2meta`" on page 59.
5. Determine which organizations will be added to the circle of trust as identity providers and service providers and create a standard and an extended metadata configuration file for each. See "Metadata" on page 38.

Note – The values in these files will come from the providers themselves.

---

6. Import the trusted provider metadata into the circle of trust as described in "Managing Metadata using `saml2meta`" on page 59.
7. Restart the web container.

Metadata

SAML profiles require that pre-interaction agreements regarding user identifiers, provider (entity) identifiers, binding support, SOAP endpoints, public key information and other similar types of data be made between providers in a circle of trust. This configuration information, or *metadata*, is defined in an XML file and shared amongst all providers who will participate in the interactions. Application programming interfaces (API) are then used to communicate with the data store; reading, writing, and managing the relevant properties and property values. There are two classifications of metadata:

- *Standard metadata* is defined in the Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0 specification. The specification includes information such as the single sign-on service URL and the assertion consumer service URL and is written so as to be extensible. For more information, see "Standard Metadata Properties" on page 39.
- *Extended metadata* are properties used by the SAML v2 Plug-in for Federation Services proprietary features and include information such as the account mapper implementation class, and the local authentication URL. For more information, see "Extended Metadata Properties" on page 41.

Instructions on how to use to the `saml2meta` command-line interface to manage metadata is in "Managing Metadata using `saml2meta`" on page 59. Instructions on how to generate a dual provider metadata configuration file is in "Dual Purpose Provider Metadata Files" on page 48.

Note – Metadata is sometimes referred to as *entity descriptor* or *entity configuration* where entity generically refers to the `entityID` with which each provider is uniquely identified. For more information on the `entityID`, see "Extended Metadata Properties" on page 41.

Standard Metadata Properties

Standard metadata properties are defined in the Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0 specification and include information such as the single sign-on service URL and the assertion consumer service URL. During installation, two standard metadata configuration files are created for use as input to the `saml2meta` utility. They are located in /*AccessManager-base*/*product-directory*/`saml2/meta` or /*FederationManager-base*/`SUNWam/saml2/meta`.

- `idpMeta.xml` is the default standard metadata configuration file if your instance of the SAML v2 Plug-in for Federation Services will act as an identity provider.
- `spMeta.xml` is the default standard metadata configuration file if your instance of the SAML v2 Plug-in for Federation Services will act as an service provider.

The following sections define both the identity provider and service provider standard metadata properties that have been implemented in the SAML v2 Plug-in for Federation Services.

- "Identity Provider Standard Metadata Properties" on page 39
- "Service Provider Standard Metadata Properties" on page 40

Note – A complete listing of all the standard metadata properties can be found in the Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0.

Identity Provider Standard Metadata Properties

The identity provider standard metadata properties implemented in the SAML v2 Plug-in for Federation Services are defined in the following table.

`WantAuthnRequestsSigned`	Takes a value of `true` or `false`. If `true`, all authentication requests received by this identity provider must be signed.
`ArtifactResolutionService`	Defines the endpoint(s) that support the Artifact Resolution profile.
`SingleLogoutService`	Defines the endpoint(s) that support the Single Logout profiles.

`ManageNameIDService`	Defines the endpoint(s) that support the Name Identifier Management profiles.
`NameIDFormat`	Defines the name identifier formats supported by the identity provider. Name identifiers are a way for providers to communicate with each other regarding a user. Single sign-on interactions support two types of identifiers: ■ A *persistent identifier* is saved to a particular user's data store entry as the value of two attributes. ■ A *transient identifier* is temporary and no data will be written to the user's persistent data store. More information about name identifiers is in "Single Sign-on" on page 63.
`SingleSignOnService`	Defines the endpoint(s) that support the profiles of the Authentication Request protocol. All identity providers must support at least one such endpoint.

Service Provider Standard Metadata Properties

The service provider standard metadata properties implemented in the SAML v2 Plug-in for Federation Services are defined in the following table.

`AuthnRequestsSigned`	Takes a value of `true` or `false`. If `true`, the service provider will sign all outgoing authentication requests.
`WantAssertionsSigned`	Takes a value of `true` or `false`. If `true`, all assertions received by this service provider must be signed.
`SingleLogoutService`	Defines the endpoint(s) that support the Single Logout profiles.
`ManageNameIDService`	Defines the endpoint(s) that support the Name Identifier Management profiles.
`NameIDFormat`	Defines the name identifier formats supported by the service provider. Name identifiers are a way for providers to communicate with each other regarding a user. Single sign-on interactions support two types of identifiers: ■ A *persistent identifier* is saved to a particular user's data store entry as the value of two attributes. ■ A *transient identifier* is temporary and no data will be written to the user's persistent data store. More information about name identifiers is in "Single Sign-on" on page 63.

`AssertionConsumerService`	Defines the endpoint(s) that support the profiles of the Authentication Request protocol. All service providers support at least one such endpoint.

Extended Metadata Properties

Extended metadata properties are properties used by our proprietary features and include information such as the account mapper implementation class and the local authentication URL. The properties are specific to whether the provider is an identity provider or a service provider. During installation, two extended metadata configuration files are created for use as input to the `saml2meta` command. They are located in `/`*AccessManager-base*`/`*product-directory*`/saml2/meta` or `/`*FederationManager-base*`/SUNWam/saml2/meta`.

- `idpExtended.xml` is the default extended metadata configuration file if your instance of the SAML v2 Plug-in for Federation Services will act as an identity provider.
- `spExtended.xml` is the default extended metadata configuration file if your instance of the SAML v2 Plug-in for Federation Services will act as an service provider.

The following sections define properties in the identity provider and service provider extended metadata.

- "Identity Provider Extended Metadata Properties" on page 41
- "Service Provider Extended Metadata Properties" on page 44

Identity Provider Extended Metadata Properties

The identity provider extended metadata properties are defined in the following table.

`hosted`	Specifies whether the entity is hosted on, or remote to, the server to which this metadata is being applied. A value of `0` or `flase` specifies that the entity is hosted. A value of `1` or `true` specifies that the entity is hosted.
`entityID`	Specifies the `EntityID` of the provider you are configuring. The value of `EntityID` for your local provider is the unique uniform resource identifier (URI) you decide to use to identity yourself to other providers. You will get a remote provider's `EntityID` from the metadata they give to you. **Note –** This `EntityID` is different from the entities configured using the console in Access Manager and Federation Manager. It is specific to SAML v2 interactions.

`metaAlias`	Specifies a `metaAlias` for the provider being configured. The `metaAlias` is used to locate the provider's entity identifier and the organization in which it is located. The value is a string equal to the realm or organization name (dependent on whether the SAML v2 Plug-in for Federation Services is installed in Access Manager or Federation Manager) coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`. **Caution** – The names used in the `metaAlias` must not contain a `/`.
`signingCertAlias`	Specifies the provider certificate alias used to find the correct signing certificate in the keystore.
`encryptionCertAlias`	Specifies the provider certificate alias used to find the correct encryption certificate in the keystore.
`basicAuthOn`	Basic authentication can be turned on to protect SOAP endpoints. This property takes a value of `true` or `false`. Any provider accessing these endpoints must have the user and password defined in the following two properties: `basicAuthUser` and `basicAuthPassword`.
`basicAuthUser`	The user associated with the basic authentication.
`basicAuthPassword`	The password associated with the basic authentication.
`autofedEnabled`	Enables auto-federation which automatically federates a user's disparate provider accounts based on a common attribute. This property takes a value of `true` or `false`.
`autofedAttribute`	Specifies the attribute used to match a user's disparate provider accounts when auto-federation is enabled.
`assertionEffectiveTime`	Specifies (in seconds) the amount of time that an assertion is valid counting from the assertion's issue time. The default value is `600` seconds.
`idpAuthncontextMapper`	Specifies the name of the implementation class for the `IDPAuthnContextMapper` interface. This class sets the authentication context in the assertion. The default value is `com.sun.identity.saml2.plugins.DefaultIDPAuthnContextMapper`, the default implementation.

`idpAuthncontextClassrefMapping`	Sets the mappings between the requested authentication context class and the actual authentication mechanism. The value of this attribute is in the format of: *authnContextClassRef* `|` *authnType*=*authnValue* `|` *authnType*=*authnValue* `|` ... where *authnContextClassRef* is the authentication context class reference, *authnType* is the module, level, or service, and *authnValue* is the module name, authentication level, or service name.
`idpAccountMapper`	Specifies the implementation of the `AccountMapper` interface used to map a remote user account to a local user account for purposes of single sign-on. The default value is `com.sun.identity.saml2.plugins.DefaultIDPAccountMapper`, the default implementation.
`idpAttributeMapper`	Specifies the implementation of the `AttributeMapper` interface used to map a remote user account attribute to a local user account attribute for purposes of single sign-on. The default value is `com.sun.identity.saml2.plugins.DefaultIDPAttributeMapper`, the default implementation
`attributeMap`	Specifies the mapping of attributes between providers. The value of this attribute is in the format: *SAML v2-attribute*=*user-attribute* where *SAML v2-attribute* is the attribute name that goes over the wire and *user-attribute* is the attribute name it will map to once it arrives. **Note –** If auto-federation is enabled, the value of *SAML v2-attribute* is equal to the value of `autofedAttribute`.
`wantNameIDEncrypted`	Takes a value of `true` or `false`. If `true`, the service provider must encrypt all `NameID` elements.
`wantArtifactResolveSigned`	Takes a value of `true` or `false`. If `true`, the service provider must sign the `ArtifactResolve` element.
`wantLogoutRequestSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `LogoutRequest` element.
`wantLogoutResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `LogoutResponse` element.
`wantMNIRequestSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ManageNameIDRequest` element.
`wantMNIResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ManageNameIDResponse` element.

`cotlist`	Specifies the name of the circle(s) of trust to which this provider belongs. As one provider may be in a number of circles, this attribute might have multiple values.

Service Provider Extended Metadata Properties

The service provider extended metadata properties are defined in the following table.

`hosted`	Specifies whether the entity is hosted on, or remote to, the server to which this metadata is being applied. A value of `0` or `flase` specifies that the entity is hosted. A value of `1` or `true` specifies that the entity is hosted.
`entityID`	Specifies the `EntityID` of the provider you are configuring. The value of `EntityID` for your local provider is the unique uniform resource identifier (URI) you decide to use to identity yourself to other providers. You will get a remote provider's `EntityID` from the metadata they give to you. **Note –** This `EntityID` is different from the entities configured using the console in Access Manager and Federation Manager. It is specific to SAML v2 interactions.
`metaAlias`	Specifies a `metaAlias` for the provider being configured. The `metaAlias` is used to locate the provider's entity identifier and the organization in which it is located. The value is a string equal to the realm or organization name (dependent on whether the SAML v2 Plug-in for Federation Services is installed in Access Manager or Federation Manager) coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`. **Caution –** The names used in the `metaAlias` must not contain a `/`.
`signingCertAlias`	Specifies the provider certificate alias used to find the correct signing certificate in the keystore.
`encryptionCertAlias`	Specifies the provider certificate alias used to find the correct encryption certificate in the keystore.
`basicAuthOn`	Basic authentication can be turned on to protect SOAP endpoints. This property takes a value of `true` or `false`. Any provider accessing these endpoints must have the user and password defined in the following two properties: `basicAuthUser` and `basicAuthPassword`.
`basicAuthUser`	The user associated with the basic authentication.
`basicAuthPassword`	The password associated with the basic authentication.

`autofedEnabled`	Auto-federation automatically federates a user's disparate provider accounts based on a common attribute. This property takes a value of `true` or `false`.
`autofedAttribute`	Specifies the attribute used to match a user's disparate provider accounts when auto-federation is enabled.
`spAccountMapper`	Specifies the implementation of the `AccountMapper` interface used to map a remote user account to a local user account for purposes of single sign-on. The default value is `com.sun.identity.saml2.plugins. DefaultSPAccountMapper`, the default implementation.
`spAttributeMapper`	Specifies the implementation of the `AttributeMapper` interface used to map a remote user account attribute to a local user account attribute for purposes of single sign-on. The default value is `com.sun.identity.saml2.plugins. DefaultSPAttributeMapper`, the default implementation
`spAuthncontextMapper`	Specifies the implementation of the `SPAuthnContextMapper` interface used to create the requested authentication context. The default implementation is `com.sun.identity.saml2.plugins. DefaultSPAttributeMapper`.
`spAuthncontextClassrefMapping`	Sets the provider's desired authentication context class and authentication level. Multiple values can be specified. The value of this property is in the format: *authnContextClassRef* `|` *authlevel* `|` *default* For example: **`urn:oasis:names:tc:SAML:2.0:ac:classes: PasswordProtectedTransport | 1`** or **`urn:oasis:names:tc:SAML:2.0:ac:classes: Password | 0 | default`**

`spAuthncontextComparisonType`	Specifies what the resulting authentication context must be when compared to the value of this property. Accepted values include: ■ *exact* where the authentication context statement in the assertion must be the exact match of, at least, one of the authentication contexts specified. ■ *minimum* where the authentication context statement in the assertion must be, at least, as strong (as deemed by the identity provider) one of the authentication contexts specified. ■ *maximum* where the authentication context statement in the assertion must be no stronger than any of the authentication contexts specified. ■ *better* where the authentication context statement in the assertion must be stronger than any of the authentication contexts specified. The default value is *exact*.
`attributeMap`	Specifies the mapping of attributes between providers. The value of this attribute is in the format: *SAML v2-attribute*=*user-attribute* where *SAML v2-attribute* is the attribute name that goes over the wire and *user-attribute* is the attribute name it will map to once it arrives. **Note –** If auto-federation is enabled, the value of *SAML v2-attribute* is equal to the value of `autofedAttribute`.
`saml2AuthModuleName`	Specifies the name of the instance of the SAML v2 authentication module. The default value is `SAML2`.
`localAuthURL`	Specifies the URL of the local login page. For more information, see "Assertion Consumer Page" on page 104.
`intermediateUrl`	Specifies a URL to which a user can be directed after authentication and before the original request's URL. An example might be a successful account creation page after the auto-creation of a user account.

`defaultRelayState`	After a successful SAML v2 operation (single sign-on, single logout, or federation termination), a page is displayed. This page, generally the originally requested resource, is specified in the initiating request using the `RelayState` element. If a `RelayState` is not specified, the value of this `defaultRelayState` property is displayed. For more information, see "Default Display Page" on page 104.
	
	Caution – When `RelayState` or `defaultRelayState` contains special characters (such as &), it must be URL-encoded. For example, if the value of `RelayState` is `http://www.sun.com/apps/myapp.jsp?param1=abc&param2=xyz`, it must be URL-encoded as:
	
	`http%3A%2F%2Fwww.sun.com%2Fapps%2Fmyapp.jsp%3Fparam1%3Dabc%26param2%3Dxyz`
	
	and then appended to the URL. For example, the service provider initiated single sign-on URL would be:
	
	`http://`*host*`:`*port*`/`*deploy-uri*`/saml2/jsp/spSSOInit.jsp?metaAlias=/sp&idpEntityID=http://www.idp.com&RelayState=http%3A%2F%2Fwww.sun.com%2Fapps%2Fmyapp.jsp%3Fparam1%3Dabc%26param2%3Dxyz`

`AssertionTimeSkew`	Assertions are valid for a period of time and not before or after. This property specifies a grace period (in seconds) for the `notBefore` value. The default value is `300`. It has no relevance to the `notAfter` value.
`wantAttributeEncrypted`	Takes a value of `true` or `false`. If `true`, the identity provider must encrypt all `AttributeStatement` elements.
`wantAssertionEncrypted`	Takes a value of `true` or `false`. If `true`, the identity provider must encrypt all `Assertion` elements.
`wantNameIDEncrypted`	Takes a value of `true` or `false`. If `true`, the identity provider must encrypt all `NameID` elements.
`wantArtifactResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ArtifactResponse` element.
`wantLogoutRequestSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `LogoutRequest` element.
`wantLogoutResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `LogoutResponse` element.
`wantMNIRequestSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ManageNameIDResponse` element.
`wantMNIResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ManageNameIDResponse` element.

`cotlist`	Specifies the name of the circle of trust to which this provider belongs.
`transientUser`	Specifies the identifier of the user to which all identity provider users will be mapped on the service provider side in cases of single sign-on using the transient name identifier.

Dual Purpose Provider Metadata Files

According to the SAML v2 specifications, one metadata file can contain configuration data for one identity provider and one service provider. Thus, it is possible to create one standard metadata configuration file and one extended configuration file which, when imported, will configure one member of a circle of trust to act as both an identity provider and a service provider. Sample files and instructions on how to generate them are found in the following sections.

- "Dual Purpose Standard Metadata Configuration File" on page 48
- "Dual Purpose Extended Metadata Configuration File" on page 50
- "To Generate Dual Purpose Metadata Configuration Files" on page 53

Dual Purpose Standard Metadata Configuration File

The dual purpose standard metadata file would contain one `<EntityDescriptor>` element containing both `<IDPSSODescriptor>` and `<SPSSODescriptor>` elements. The following sample is a standard metadata configuration file in which the data configures `zosma21.central.sun.com` as both a service provider and an identity provider.

```
<EntityDescriptor
    xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
    entityID="zosma21.central.sun.com/">
    <IDPSSODescriptor
        WantAuthnRequestsSigned="false"
        protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <ArtifactResolutionService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
            Location="http://zosma21.central.sun.com:80/amserver/ArtifactResolver/
             metaAlias/idp"
            index="0"
            isDefault="1"/>
        <SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
            Location="http://zosma21.central.sun.com:80/amserver/IDPSloRedirect/
             metaAlias/idp"
            ResponseLocation="http://zosma21.central.sun.com:80/amserver/
             IDPSloRedirect/metaAlias/idp"/>
        <SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
```

```
                Location="http://zosma21.central.sun.com:80/amserver/
                 IDPSloSoap/metaAlias/idp"/>
            <ManageNameIDService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
                Location="http://zosma21.central.sun.com:80/amserver/IDPMniRedirect/
                 metaAlias/idp"
                ResponseLocation="http://zosma21.central.sun.com:80/amserver/
                 IDPMniRedirect/metaAlias/idp"/>
            <ManageNameIDService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                Location="http://zosma21.central.sun.com:80/amserver/IDPMniSoap/
                 metaAlias/idp"/>
            <NameIDFormat>
                urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
            </NameIDFormat>
            <NameIDFormat>
                urn:oasis:names:tc:SAML:2.0:nameid-format:transient
            </NameIDFormat>
            <SingleSignOnService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
                Location="http://zosma21.central.sun.com:80/amserver/SSORedirect/
                 metaAlias/idp"/>
            <SingleSignOnService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                Location="http://zosma21.central.sun.com:80/amserver/SSOSoap/
                 metaAlias/idp"/>
        </IDPSSODescriptor>
        <SPSSODescriptor
            AuthnRequestsSigned="false"
            WantAssertionsSigned="false"
            protocolSupportEnumeration=
                "urn:oasis:names:tc:SAML:2.0:protocol">
            <SingleLogoutService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
                Location="http://zosma21.central.sun.com:80/amserver/SPSloRedirect/
                 metaAlias/sp"
                ResponseLocation="http://zosma21.central.sun.com:80/amserver/
                 SPSloRedirect/metaAlias/sp"/>
            <SingleLogoutService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                Location="http://zosma21.central.sun.com:80/amserver/SPSloSoap/
                 metaAlias/sp"/>
            <ManageNameIDService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
                Location="http://zosma21.central.sun.com:80/amserver/SPMniRedirect/
                 metaAlias/sp"
                ResponseLocation="http://zosma21.central.sun.com:80/amserver/
                 SPMniRedirect/metaAlias/sp"/>
        <ManageNameIDService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
            Location="http://zosma21.central.sun.com:80/amserver/SPMniSoap/
             metaAlias/sp"
            ResponseLocation="http://zosma21.central.sun.com:80/amserver/
             SPMniSoap/metaAlias/sp"/>
        <NameIDFormat>
            urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
        </NameIDFormat>
        <NameIDFormat>
            urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <AssertionConsumerService
            isDefault="true"
            index="0"
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
            Location="http://zosma21.central.sun.com:80/amserver/Consumer/
             metaAlias/sp"/>
        <AssertionConsumerService
            index="1"
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
            Location="http://zosma21.central.sun.com:80/amserver/Consumer/
             metaAlias/sp"/>
    </SPSSODescriptor>
</EntityDescriptor>
```

Dual Purpose Extended Metadata Configuration File

The dual purpose extended metadata file would contain one `<EntityConfig>` element containing both `<IDPSSOConfig>` and `<SPSSOConfig>` elements. The following sample is an extended metadata configuration file in which the data configures `zosma21.central.sun.com` as both a service provider and an identity provider.

```
<EntityConfig xmlns="urn:sun:fm:SAML:2.0:entityconfig"
    xmlns:fm="urn:sun:fm:SAML:2.0:entityconfig"
    hosted="1"
    entityID="zosma21.central.sun.com/">
    <IDPSSOConfig metaAlias="/idp">
       <Attribute name="signingCertAlias">
           <Value></Value>
       </Attribute>
       <Attribute name="encryptionCertAlias">
           <Value></Value>
       </Attribute>
       <Attribute name="basicAuthOn">
           <Value>false</Value>
       </Attribute>
       <Attribute name="basicAuthUser">
    <Value></Value>
</Attribute>
<Attribute name="basicAuthPassword">
    <Value></Value>
</Attribute>
<Attribute name="autofedEnabled">
    <Value>false</Value>
</Attribute>
<Attribute name="autofedAttribute">
    <Value></Value>
</Attribute>
<Attribute name="assertionEffectiveTime">
    <Value>600</Value>
</Attribute>
<Attribute name="idpAuthncontextMapper">
    <Value>com.sun.identity.saml2.plugins.DefaultIDPAuthnContextMapper</Value>
</Attribute>
<Attribute name="idpAuthncontextClassrefMapping">
    <Value>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</Value>
</Attribute>
<Attribute name="idpAccountMapper">
    <Value>com.sun.identity.saml2.plugins.DefaultIDPAccountMapper</Value>
</Attribute>
<Attribute name="idpAttributeMapper">
    <Value>com.sun.identity.saml2.plugins.DefaultIDPAttributeMapper</Value>
</Attribute>
<Attribute name="attributeMap">
    <Value></Value>
</Attribute>
<Attribute name="wantNameIDEncrypted">
    <Value></Value>
</Attribute>
<Attribute name="wantArtifactResolveSigned">
    <Value></Value>
</Attribute>
<Attribute name="wantLogoutRequestSigned">
    <Value></Value>
</Attribute>
<Attribute name="wantLogoutResponseSigned ">
    <Value></Value>
</Attribute>
<Attribute name="wantMNIRequestSigned">
    <Value></Value>
</Attribute>
<Attribute name="wantMNIResponseSigned">
    <Value></Value>
</Attribute>
<Attribute name="cotlist">
```

```
      </Attribute>
   </IDPSSOConfig>
   <SPSSOConfig metaAlias="/sp">
      <Attribute name="signingCertAlias">
          <Value></Value>
      </Attribute>
      <Attribute name="encryptionCertAlias">
          <Value></Value>
      </Attribute>
      <Attribute name="basicAuthOn">
          <Value>false</Value>
      </Attribute>
      <Attribute name="basicAuthUser">
          <Value></Value>
      </Attribute>
      <Attribute name="basicAuthPassword">
          <Value></Value>
      </Attribute>
      <Attribute name="autofedEnabled">
          <Value>false</Value>
      </Attribute>
      <Attribute name="autofedAttribute">
          <Value></Value>
      </Attribute>
      <Attribute name="transientUser">
          <Value></Value>
      </Attribute>
      <Attribute name="spAccountMapper">
          <Value>com.sun.identity.saml2.plugins.DefaultSPAccountMapper</Value>
      </Attribute>
      <Attribute name="spAttributeMapper">
          <Value>com.sun.identity.saml2.plugins.DefaultSPAttributeMapper</Value>
      </Attribute>
      <Attribute name="spAuthncontextMapper">
          <Value>com.sun.identity.saml2.plugins.DefaultSPAuthnContextMapper</Value>
      </Attribute>
      <Attribute name="spAuthncontextClassrefMapping">
          <Value>PasswordProtectedTransport|0|default</Value>
      </Attribute>
      <Attribute name="spAuthncontextComparisonType">
          <Value>exact</Value>
      </Attribute>
      <Attribute name="attributeMap">
          <Value></Value>
      </Attribute>
      <Attribute name="saml2AuthModuleName">
          <Value></Value>
      </Attribute>
```

```
        <Attribute name="localAuthURL">
            <Value></Value>
        </Attribute>
        <Attribute name="intermediateUrl">
            <Value></Value>
        </Attribute>
        <Attribute name="defaultRelayState">
            <Value></Value>
        </Attribute>
        <Attribute name="assertionTimeSkew">
            <Value>300</Value>
        </Attribute>
        <Attribute name="wantAttributeEncrypted">
            <Value></Value>
        </Attribute>
        <Attribute name="wantAssertionEncrypted">
            <Value></Value>
        </Attribute>
        <Attribute name="wantNameIDEncrypted">
            <Value></Value>
        </Attribute>
        <Attribute name="wantArtifactResponseSigned">
            <Value></Value>
        </Attribute>
        <Attribute name="wantLogoutRequestSigned">
            <Value></Value>
        </Attribute>
        <Attribute name="wantLogoutResponseSigned ">
            <Value></Value>
        </Attribute>
        <Attribute name="wantMNIRequestSigned">
            <Value></Value>
        </Attribute>
        <Attribute name="wantMNIResponseSigned">
            <Value></Value>
        </Attribute>
        <Attribute name="cotlist">
        </Attribute>
    </SPSSOConfig>
</EntityConfig>
```

▼ To Generate Dual Purpose Metadata Configuration Files

This procedure creates one standard metadata file and one extended metadata file that contains configuration information for one provider that, when imported, will define it as capable of both functions. See "The saml2meta Command-line Reference" on page 57 for more information on the saml2meta command line interface.

1. **Generate the dual purpose standard and extended metadata configuration files.**

   saml2meta [-i *staging-directory*] **template -u amadmin -w** *password* **-e dual -s /sp1 -d /idp1 -m dualMeta.xml -x dualExtended.xml**

2. **Import the generated standard and extended metadata configuration files.**

   saml2meta [-i *staging-directory*] **import -u amadmin -w** *password* **-m dualMeta.xml -x dualExtended.xml**

Circles of Trust

Circles of trust need to be created to define trust relationships among identity providers and service providers. A *circle of trust* is a grouping of service providers (with at least one identity provider) that have, in place, pertinent business agreements regarding how they will do business and interact with identities. Any identity provider or service provider within a circle of trust will honor requests and information that come from other providers in the same circle. Requests and information will be rejected if communicating providers are not part of the same circle of trust. In the SAML v2 Plug-in for Federation Services, circles of trust are created using the `saml2meta` command-line interface, allowing you to configure technologically the participants and rules drawn in the business agreements. Instructions on how to use the `saml2meta` command-line interface to manage circles of trust is in "Managing Circles of Trust using `saml2meta`" on page 60.

AMConfig.properties

`AMConfig.properties` is the main configuration file for Access Manager and Federation Manager. `AMConfig.properties` is located in the `/etc/opt/`*product-directory*`/config` directory in Access Manager and the `/`*staging-directory*`/web-src/WEB-INF/classes` directory in Federation Manager.

Static Properties in AMConfig.properties

This list describes the properties added to `AMConfig.properties` when the SAML v2 Plug-in for Federation Services is installed.

Caution – Do not modify the properties configured during installation.

- **`com.sun.identity.saml2.am_or_fm`** takes a value of `AM` for Access Manager or `FM` for Federation Manager. It specifies the instance type onto which the SAML v2 Plug-in for Federation Services is installed.

- **`com.sun.identity.saml2.xmlenc.EncProviderImpl=`** **`com.sun.identity.saml2.xmlenc.FMEncProvider`** specifies the XML encryption provider implementation class.
- **`com.sun.identity.saml2.xmlenc.SigProviderImpl=`** **`com.sun.identity.saml2.xmlsig.FMSigProvider`** specifies the XML signature provider implementation class.
- **`com.sun.identity.common.datastore.provider.default=`** **`com.sun.identity.saml2.plugins.IdRepoDataStoreProvider`** specifies the data store provider implementation class. The `IdRepoDataStoreProviderdefault` class provides implementation using the identity repository API.

  Note – In Federation Manager, a different implementation class is already set. It is not set by the SAML v2 Plug-in for Federation Services installer.

- The **`com.sun.identity.saml2.nameidinfo.attribute`** and **`com.sun.identity.saml2.nameidinfokey`** properties specify the LDAPv3 attribute to which you want federation information written in a principal's account. For more information, see "Using Non-Default Federation Attributes" on page 70.

Additional Properties in `AMConfig.properties`

This list describes the properties and accompanying values that may be added to `AMConfig.properties` after installation.

- **`com.sun.identity.saml2.cacheCleanUpInterval`** takes a value in seconds. It specifies the amount of time between each cache cleanup.
- **`com.sun.identity.saml2.sdk.mapping.`***interface-name*=*new-class-name* is a mapping property for customized implementation classes. For more information, see "The SAML v2 Plug-in for Federation Services SDK" on page 93.

The SAML v2 IDP Discovery Service

The SAML v2 IDP Discovery Service is an implementation of the Identity Provider Discovery Profile as described in the Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0 specification. In deployments having more than one identity provider, service providers need to determine which identity provider(s) a principal uses with the Web Browser SSO profile. To allow for this, the SAML v2 IDP Discovery Service relies on a cookie written in a domain that is common to all identity providers and service providers in a circle of trust. This predetermined domain is known as the *common domain*, and the cookie containing the list of identity providers to chose from is known as the *common domain cookie*.

When a user requests access from a service provider and an entity identifier for an identity provider is not received in the request, the service provider redirects the request to the common domain's SAML v2 IDP Discovery Service Reader URL to retrieve the identity provider's entity identifier. If more then one identity provider entity identifier is returned, the last entity identifier in the list is the one to which the request is redirected. Once received, the identity provider redirects to the Discovery Service Writer URL to set the common domain cookie using the value defined in the installation configuration properties file. See "Creating an Installation Configuration Properties File" on page 28 for more information.

Note – The Reader and Writer URLs for the SAML v2 IDP Discovery Service are defined when configuring the circle of trust. If the circle already exists and does not contain values for the Reader and Writer URLs, it must be deleted and recreated.

▼ To Set the Reader and Writer URLs

Before You Begin Instructions on how to install the SAML v2 IDP Discovery Service can be found in "Installing the SAML v2 IDP Discovery Service" on page 32. You should also be familiar with "The `saml2meta` Command-line Reference" on page 57 as well as Table 3–2.

1. **Delete the current circle of trust configuration using `saml2meta`, if applicable.**

2. **Create a new circle of trust configuration using `saml2meta` and the `cotcreate` subcommand.**

   ```
   saml2meta [-i staging-directory] cotcreate -u admin-user -w password -t COT-name
    -p idp-discovery-URL-path
   ```

   Make sure to specify the full path to where the SAML v2 Plug-in for Federation Services is deployed using the `-p` option.

3. **Add member providers to the new circle of trust using `saml2meta` and the `cotadd` subcommand.**

   ```
   saml2meta [-i staging-directory] cotadd -u admin-user -w password -t COT-name -e entity-ID
   ```

Tip – `cotadd` can only add a single provider at a time using the `-e` option. To add a group of providers, you can use the `-l` option with `cotcreate` in the previous step.

---

4. **Verify that all member providers have been added to the circle using `saml2meta` and the `cotlist` subcommand.**

   ```
   saml2meta [-i staging-directory] cotlist -u admin-user -w password
   ```

Next Steps Service providers will be redirected to the SAML v2 IDP Discovery Service Reader URL during single sign-on.

The `saml2meta` Command-line Reference

The SAML v2 Plug-in for Federation Services contains the `saml2meta` command-line interface to manage metadata and circles of trust. It is located in `/`*AccessManager-base*`/`*product-directory*`/saml2/meta` or `/`*FederationManager-base*`/SUNWam/saml2/meta`.

The `saml2meta` syntax is:

```
saml2meta [-i staging-directory] import -u user-DN [-w password | -j password-file] [-r realm]
 [-m XML-file-name] [-x XML-file-name] [-t COT_name]

saml2meta [-i staging-directory] export -u user-DN [-w password | -j password-file] [-r realm]
 -e entityID [-n] [ -m  XML-file-name] [-x XML-file-name]

saml2meta [-i staging-directory] template -u user-DN [-w password | -j password-file]
 [-e entityID] [-s metaAlias [-a certAlias] [-f certAlias]] [-d metaAlias
 [-b certAlias] [-g certAlias]] -m XML-file-name -x XML-file-name

saml2meta [-i staging-directory] delete -u user-DN [-w password | -j password-file]
 [-r realm] [-e entityID] [-c]

saml2meta [-i staging-directory] list -u user-DN [-w password | -j password-file]

saml2meta [-i staging-directory] cotcreate -u user-DN [-w password | -j password-file]
 [-t COT-name] [-p prefix-URL] [-l entity-ID, entity-ID, ...]

saml2meta [-i staging-directory] cotdelete -u user-DN [-w password | -j password-file]
 [-t COT-name]

saml2meta [-i staging-directory] cotadd -u user-DN [-w password | -j password-file]
 [-t COT-name] [-e entityID]

saml2meta [-i staging-directory] cotremove -u user-DN [-w password | -j password-file]
 [-t COT-name] [-e entityID]

saml2meta [-i staging-directory] cotmember -u user-DN [-w password | -j password-file]
 -t COT-name

saml2meta [-i staging-directory] cotlist -u user-DN [-w password | -j password_file]

saml2meta -V

saml2meta -?
```

where:

`-i`	Specifies the directory for the web application staging area. For example, `/var/opt/SUNWam/fm/war-staging` on Federation Manager. **Note –** This option is specific only to instances of the SAML v2 Plug-in for Federation Services installed on Federation Manager.
`-u` or `--runasdn`	Specifies the full distinguished name of the user running the command.
`-w` or `--password`	Specifies the password of the user running the command.
`-j` or `--passwordfile`	Specifies the name of the file that contains the password of the user running the command.
`-r` or `--realm`	Specifies the realm or organization under which the metadata is stored. If not defined, the default value is the root realm or organization.
`-m` or `--metadata`	Specifies a file name for the standard configuration. **Note –** In most subcommands, either `-m` or `-x` must be used. Both can also be used.
`-x` or `--extended`	Specifies a file name for the extended configuration. **Note –** In most subcommands, either `-m` or `-x` must be used. Both can also be used.
`-e` or `--entityid`	Specifies an entity identifier, if applicable.
`-s` or `--serviceprovider`	Specifies a `metaAlias` for the hosted service provider being created. The `metaAlias` is used to locate the provider's entity identifier and the organization in which it is located. The value is a string equal to the realm or organization name (dependent on whether the SAML v2 Plug-in for Federation Services is installed in Access Manager or Federation Manager) coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`. **Caution –** The strings used in the `metaAlias` values must not contain a `/`.
`-a` or `--spcertalias`	Specifies a certificate alias for the hosted service provider to be created.
`-f` or `--specertalias`	Specifies an encrypted certificate alias for the hosted service provider to be created.
`-d` or `--identityprovider`	Specifies a `metaAlias` for the hosted identity provider being created. The `metaAlias` is used to locate the provider's entity identifier and the organization in which it is located. The value is a string equal to the realm or organization name (dependent on whether the SAML v2 Plug-in for Federation Services is installed in Access Manager or Federation Manager) coupled with a forward slash and the provider name. For example, `/suncorp/hr`. **Caution –** The strings used in the `metaAlias` values must not contain a `/`.
`-b` or `--idpcertalias`	Specifies a certificate alias for the hosted identity provider to be created.

-g or --idpecertalias	Specifies an encrypted certificate alias for the hosted identity provider to be created.
-n or --sign	Signs the exported XML file.
-c or --extendedonly	Deletes extended configurations only.
-t or --cot	Specifies the name of a circle of trust.
-p or --prefix	Specifies the full path to where the SAML v2 IDP Discovery Service is deployed.
-l or --trustedproviders	Specifies a list of trusted providers in a circle of trust. Input is a comma-separated list of entity identifiers.
-V	Displays version information.
-?	Displays help information.

Note – To access usage information on the command-line, change to /*AccessManager-base*/*product-directory*/saml2/bin or /*FederationManager-base*/SUNWam/saml2/bin and run saml2meta with no input.

For explanations of the saml2meta subcommands, see the:

- saml2meta Subcommands for Managing Metadata in Table 3–1
- saml2meta Subcommands for Managing Circles of Trust in Table 3–2

Managing Metadata using saml2meta

saml2meta is used to manage the SAML v2 metadata. The following table describes the saml2meta subcommands specific to metadata management.

TABLE 3–1 saml2meta Subcommands for Managing Metadata

Subcommand	Description
import	Loads standard and extended metadata in XML format into a local configuration data store. **Note –** Either -m or -x must be used. Both can also be used.
export	Exports standard and extended metadata in XML format from a local configuration data store. **Note –** Either -m or -x must be used. Both can also be used.

TABLE 3–1 saml2meta Subcommands for Managing Metadata *(Continued)*

Subcommand	Description
`template`	Generates a metadata configuration file for either type of hosted provider (service or identity) with defined values for default metadata properties. The generated file can be modified for use with `import`.
`delete`	Removes standard or extended metadata from a local configuration data store.
`list`	Generates a list of all the entity identifiers on the system.

Following are some examples on how you might use `saml2meta`. See "The `saml2meta` Command-line Reference" on page 57 for explanations of the options used.

- The following command example will create both a standard and an extended metadata configuration file for service provider `sp.sun.com`:

  ```
  # saml2meta template -u amadmin -w password -e sp.sun.com -s /sp
   -m spMeta.xml -x spExtended.xml
  ```

  The standard metadata is defined in `spMeta.xml` and the extended metadata is defined in `spExtended.xml`.

- This command example will import the created files into the local configuration data store:

  ```
  # saml2meta import -u amadmin -w password -m spMetadata.xml -x spExtended.xml
  ```

  Note – Remember to delete old metadata before you import modified files.

- See "Dual Purpose Provider Metadata Files" on page 48.

Managing Circles of Trust using `saml2meta`

The `saml2meta` command line interface creates and manages the circles of trust used by the SAML v2 Plug-in for Federation Services. The following table describes the `saml2meta` subcommands specific to circle of trust management.

TABLE 3–2 saml2meta Subcommands for Managing Circles of Trust

Subcommand	Description
`cotcreate`	Creates a circle of trust.

TABLE 3–2 saml2meta Subcommands for Managing Circles of Trust *(Continued)*

Subcommand	Description
cotdelete	Removes a circle of trust. **Note –** To delete a circle of trust that contains providers, use cotremove to remove each provider first, then use cotdelete to delete the circle itself.
cotadd	Adds a trusted provider to an existing circle of trust. **Note –** cotadd can only add a single entity at a time. Add multiple entities when you first create the circle by using cotcreate and the -l option.
cotremove	Removes a trusted provider from an existing circle of trust.
cotmember	Lists the member providers in a particular circle of trust.
cotlist	Lists all the circles of trust configured on the system.

The following command example will create a circle of trust:

```
saml2meta [-i staging-directory] cotcreate -u admin-user -w password -t COT-name
 -p idp-discovery-URL-path
```

This second command example will add a trusted provider to an existing circle of trust:

```
saml2meta [-i staging-directory] cotadd -u admin-user -w password -t COT-name -e entity-ID
```

This next command example will remove a trusted provider from an existing circle of trust:

```
saml2meta [-i staging-directory] cotremove -u admin-user -w password -t COT-name -e entity-ID
```

This command example will list all the providers belonging to an existing circle of trust:

```
saml2meta [-i staging-directory] cotmember -u admin-user -w password -t COT-name
```

This last command example will list all the available circles of trust under the instance of the SAML v2 Plug-in for Federation Services:

```
saml2meta [-i staging-directory] cotlist -u admin-user -w password
```

CHAPTER 4

Configuring Specialized Interactions

The SAML v2 Plug-in for Federation Services can be configured to support many specialized interactions. The use cases described in this chapter give you a general idea of how the SAML v2 Plug-in for Federation Services can be used.

This chapter covers the following topics:

Single Sign-on

Name identifiers are used by the identity provider and the service provider to communicate with each other regarding a user. Single sign-on interactions can support both persistent and transient identifiers. A *persistent identifier* is saved to a particular user entry as the value of two attributes. A *transient identifier* is temporary and no data will be written to the user's data store entry.

In some deployments, there might be no user account on the service provider side of an interaction. In this case, single sign-on with the transient name identifier is used. All users passed from the identity provider to the service provider will be mapped to this one user account. All attributes defined in the `AttributeStatement` will be set as properties of the specific user's single sign-on token. The following procedures describe some interactions using the transient name identifier.

- "To Use the Transient Name Identifier" on page 64
- "To Configure Single Sign-on without Service Provider User Account" on page 64

▼ To Use the Transient Name Identifier

1. **Append the `NameIDFormat=transient` query parameter to the URL that initiates a single sign-on JavaServer Page™ (JSP™).**

   `spSSOInit.jsp` and `idpSSOInit.jsp` both initiate single sign-on interactions.

2. **To test, invoke the URL.**

   For more information, see "JavaServer Pages" on page 103.

▼ To Configure Single Sign-on without Service Provider User Account

In some deployments, the service provider side of an interaction might not store user accounts. The single sign-on solution is for all identity provider user accounts to be mapped to one service provider user account. Any attributes inside the `AttributeStatement` will be set as properties of the single sign-on token. The following procedure maps an identity provider user to a service provider `anonymous` user and passes two attributes to the service provider.

1. **Export the identity provider's current extended metadata configuration to a file.**

   `saml2meta [-i` *staging-directory*] **`export -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-x`** *IDP-extended-XML-file-name*

2. **Edit `attributeMap` in the exported extended metadata configuration file.**

   `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute-value*=*remote-provider-attribute*. For example:

```
<Attribute name="attributeMap">
<Value>mail=mail</Value>
<Value>employeeNumber=employeeNumber</Value>
</Attribute>
```

3. **Remove the identity provider's current extended metadata configuration.**

   `saml2meta [-i` *staging-directory*] **`delete -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-c`**

4. **Import the identity provider's modified extended metadata configuration file.**

   `saml2meta [-i` *staging-directory*] **`import -u amadmin -w`** *password* **`-x`** *IDP-extended-XML-file-name*

5 **Restart the web container.**

6 **Export the service provider's current extended metadata configuration to a file.**

saml2meta [-i *staging-directory*] **export -u amadmin -w** *password* **-e** *SP-entityID* **-x** *SP-extended-XML-file-name*

7 **Edit the following attributes in the exported extended metadata configuration file.**

- `transientUser` will take a value of one of the existing transient user identifiers on the service provider side, for example, `anonymous`.
- `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute_value*=*remote_provider_attribute*. For example:

```
<Attribute name="attributeMap">
<Value>mail=mail</Value>
<Value>employeeNumber=employeeNumber</Value>
</Attribute>
```

8 **Remove the service provider's current extended metadata configuration.**

saml2meta [-i *staging-directory*] **delete -u amadmin -w** *password* **-e** *SP-entityID* **-c**

9 **Import the service provider's modified extended metadata configuration file.**

saml2meta [-i *staging-directory*] **import -u amadmin -w** *password* **-x** *SP-extended-XML-file-name*

10 **Restart the web container.**

11 **To test, invoke the single sign-on URL with the `NameIDFormat=transient` query parameter appended to it.**

All identity provider users will be mapped to `anonymous` on the service provider side. `mail` and `employeeNumber` will be set as properties in the identity provider user's single sign-on token. For more information on the single sign-on URL, see "JavaServer Pages" on page 103.

Auto-Federation

The auto-federation feature will automatically federate a user's disparate provider accounts based on a common attribute defined in the interacting provider's metadata. (It is also referred to as *attribute federation*.) This common attribute, when exchanged in a single sign-on assertion, would identify the user at both provider sites and automatically create the appropriate federations. The following sections describe procedures for auto-federation.

- "To Enable Auto-Federation" on page 66

- "To Configure Single Sign-on Without Data Store Writes" on page 67

Note – Auto-federation with the transient name identifier can also be configured as described in "To Configure Single Sign-on without Service Provider User Account" on page 64.

▼ To Enable Auto-Federation

Before You Begin You must configure the attribute mapper on the identity provider side to include the common attribute as part of the `AttributeStatement`. You must also configure the attribute mapper on the service provider side to use the common attribute to find the user.

Note – You can also configure the account mapper on the service provider side to map all users to a single user (such as `anonymous`).

---

1 **Export the identity provider's current extended metadata configuration to a file.**

`saml2meta [-i` *staging-directory***`] export -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-x`** *IDP-extended-XML-file-name*

2 **Edit the following attributes in the exported extended metadata configuration file.**

- `autofedEnabled` takes a value of `true`.
- `autofedAttribute` defines the common attribute. For example, `<Value>employeeNumber</Value>`
- `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute-value***=***remote-provider-attribute*. For example:

```
<Attribute name="attributeMap">
<Value>employeeNumber=employeeNumber</Value>
</Attribute>
```

3 **Remove the identity provider's current extended metadata configuration.**

`saml2meta [-i` *staging-directory***`] delete -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-c`**

4 **Import the identity provider's modified extended metadata configuration file.**

`saml2meta [-i` *staging-directory***`] import -u amadmin -w`** *password* **`-x`** *IDP-extended-XML-file-name*

5 **Restart the web container.**

6 **Repeat the above steps to modify the service provider's extended metadata.**

7 **To test, invoke single sign-on from the service provider.**

Following the auto-federation, two SAML v2 attributes and corresponding values are written to the user's data store entry.

▼ To Configure Single Sign-on Without Data Store Writes

This interaction uses auto-federation with the transient name identifier. There is one-to-one mapping between user accounts configured with the identity provider and the service provider based on the value of one attribute. The following procedure describes how to configure single sign-on without writing to the user's data store entry.

1 **Export the identity provider's current extended metadata configuration to a file.**

`saml2meta [-i` *staging-directory*] **`export -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-x`** *IDP-extended-XML-file-name*

2 **Edit the following attributes in the exported extended metadata configuration file.**

- `autofedEnabled` takes a value of `true`.
- `autofedAttribute` defines the common attribute on the identity provider side. For example, `mail`.
- `attributeMap` defines the mapping between the identity provider's attribute and the remote provider's attribute. It takes a value of *autofedAttribute-value***=***remote-provider-attribute*. For example:

```
<Attribute name="attributeMap">
<Value>mail=mail</Value>
</Attribute>
```

3 **Remove the identity provider's current extended metadata configuration.**

`saml2meta [-i` *staging-directory*] **`delete -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-c`**

4 **Import the identity provider's modified extended metadata configuration file.**

`saml2meta [-i` *staging-directory*] **`import -u amadmin -w`** *password* **`-x`** *IDP-extended-XML-file-name*

5 **Restart the web container.**

6 **Export the service provider's current extended metadata configuration to a file.**

`saml2meta [-i` *staging-directory*] **`export -u amadmin -w`** *password* **`-e`** *SP-entityID* **`-x`** *SP-extended-XML-file-name*

7 **Edit the following attributes in the exported extended metadata configuration file.**

- `transientUser` takes a null value.
- `autofedEnabled` takes a value of `true`.
- `autofedAttribute` defines the common attribute. For example, `mail`.
- `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute-value*=*remote-provider-attribute*. For example:

```
<Attribute name="attributeMap">
<Value>mail=mail</Value>
</Attribute>
```

8 **Remove the service provider's current extended metadata configuration.**

saml2meta [-i *staging-directory***] delete -u amadmin -w** *password* **-e** *SP-entityID* **-c**

9 **Import the service provider's modified extended metadata configuration file.**

saml2meta [-i *staging-directory***] import -u amadmin -w** *password* **-x** *SP-extended-XML-file-name*

10 **Restart the web container.**

11 **To test, invoke the single sign-on URL with the `NameIDFormat=transient` query parameter appended to it.**

All identity provider users will be mapped to the corresponding user on the service provider side based on the `mail` attribute but the auto-federation attributes will not be written to the user entry.

Auto-creation of User Accounts

Auto-creation of user accounts can be enabled on the service provider side. An account would be created when there is none corresponding to the identity provider user account requesting access. This might be necessary, for example, when a new service provider has joined an existing circle of trust.

Note – Auto-creation is supported only when the service provider is running on an instance of Access Manager as it extends that product's Dynamic Profile Creation functionality.

▼ To Enable Auto-creation

Before You Begin You must configure the attribute mapper on the identity provider side to include an `AttributeStatement` from the user. The account mapper on the service provider side will perform user mapping based on the `AttributeStatement`.

1. **Export the identity provider's current extended metadata configuration to a file.**

   `saml2meta [-i` *staging-directory*`]` **`export -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-x`** *IDP-extended-XML-file-name*

2. **Edit the following attributes in the exported extended metadata configuration file.**

   - `autofedEnabled` takes a value of `true`.
   - `autofedAttribute` defines the common attribute. For example, `<Value>employeeNumber</Value>`
   - `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute-value*=*remote-provider-attribute*. For example:

     ```
     <Attribute name="attributeMap">
     <Value>employeeNumber=employeeID</Value>
     </Attribute>
     ```

3. **Remove the identity provider's current extended metadata configuration.**

   `saml2meta [-i` *staging-directory*`]` **`delete -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-c`**

4. **Import the identity provider's modified extended metadata configuration file.**

   `saml2meta [-i` *staging-directory*`]` **`import -u amadmin -w`** *password* **`-x`** *IDP-extended-XML-file-name*

5. **Restart the web container.**

6. **Repeat the above steps to modify the service provider's extended metadata.**

7. **Enable Dynamic Profile Creation using the Access Manager console.**

   a. **Log in to the Access Manager console as the top-level administrator, by default,** `amadmin`**.**

   b. **Under the Access Control tab, select the appropriate realm.**

   c. **Select the Authentication tab.**

   d. **Select Advanced Properties.**

e. **Set User Profile to Dynamic or Dynamic with User Alias and click Save.**

f. **Log out of Access Manager.**

8 **To test, invoke single sign-on from the service provider.**

For more information, see the *Sun Java System Access Manager 7 2005Q4 Administration Guide*.

Using Non-Default Federation Attributes

If Access Manager or Federation Manager is retrieving data from an LDAPv3–compliant directory, the object class `sunFMSAML2NameIdentifier` (containing two allowed attributes, `sunfm- saml2-nameid-info` and `sun-fm-saml2-nameid-infokey`) needs to be loaded into the entries of all existing users. When the directory contains a large user database the process is time-intensive. The following procedure can be used to modify your SAML v2 Plug-in for Federation Services installation to use existing LDAP attributes to store user federation information. In this case, there is no need to change the schema.

▼ To Store Federation Information in Existing Attributes

1 **Modify the values of the following properties in `AMConfig.properties` to reflect the existing attributes to which you want federation information written:**

- `com.sun.identity.saml2.nameidinfo.attribute`
- `com.sun.identity.saml2.nameidinfokey.attribute`

Note – `AMConfig.properties` is located in the `/etc/opt/`*product-directory*`/config` directory in Access Manager and in the `/`*staging-directory*`/web-src/WEB-INF/classes` directory in Federation Manager.

---

2 **Restart the web container.**

Federation information will now be written to the specified attributes.

Enabling XML Signing and Encryption

This section contains the procedure to enable XML signing and encryption.

Caution – If you are modifying your organization's metadata remember to contact all providers in the circle of trust with the new metadata.

▼ To Enable XML Signing and Encryption

1 **In `AMConfig.properties`, set `com.sun.identity.jss.donotInstallAtHighestPriority` equal to `true`.**

Note – `AMConfig.properties` is located in the `/etc/opt/`*product-directory*`/config` directory in Access Manager and in the `/`*staging-directory*`/web-src/WEB-INF/classes` directory in Federation Manager.

2 **Follow the instructions in the XMLSIG sample to setup a keystore and import the signing and encryption certificates to the keystore.**

In Access Manager, the sample is located in the `/`*AccessManager-base*`/`*product-directory*`/samples/saml/xmlsig` directory. In Federation Manager, the sample is located in the `/`*FederationManager-base*`/SUNWam/fm/samples/saml/xmlsig` directory.

Note – The certificate alias assigned during this process will be used in the following steps to identify the certificate.

3 **Regenerate the metadata files so that they include the signing and encryption key information.**

- For identity provider metadata, run the following command:

  `saml2meta template [-i` *war-staging*`] -u` *admin* `-w` *admin-password* `-d` *idp-metaAlias* `-b` *idp-signing-key-alias* `-g` *idp-encryption-key-alias* `-e` *idp-entityID* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name*

  For example:

  ```
  saml2meta template -u amadmin -w 11111111 -d /idp -b test -g test -e
  idp.sun.com -m idpMeta.xml -x idpExt.xml
  ```

- For service provider metadata, run the following command:

  `saml2meta template [-i` *war-staging*`] -u` *admin* `-w` *admin-password* `-s` *sp-metaAlias* `-a` *sp-signing-key-alias* `-f` *sp-encryption-key-alias* `-e` *sp-entityID* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name*

  For example:

  ```
  saml2meta template -u amadmin -w 11111111 -s /idp -a test -f test -e
  sp.sun.com -m spMeta.xml -x spExt.xml
  ```

4 Enable the appropriate XML signing and encryption features by modifying the generated metadata files.

Note – XML signing is required for the Web Browser POST Profile.

You can turn on XML signing and encryption features by changing the value of the following attributes to `true`:

- Identity Provider Standard Metadata Configuration File Attribute
  - `wantAuthnRequestsSigned`
- Service Provider Standard Metadata Configuration File Attributes
  - `AuthnRequestsSigned`
  - `WantAssertionsSigned`
- Identity Provider Extended Metadata Configuration File Attributes
  - `wantNameIDEncrypted`
  - `wantArtifactResolveSigned`
  - `WantLogoutRequestSigned`
  - `WantLogoutResponseSigned`
  - `WantMNIRequestSigned`
  - `WantMNIResponseSigned`
- Service Provider Extended Metadata Configuration File Attributes
  - `wantAttributeEncrypted`
  - `wantAssertionEncrypted`
  - `wantNameIDEncrypted`
  - `wantArtifactResponseSigned`
  - `WantLogoutRequestSigned`
  - `WantLogoutResponseSigned`
  - `WantMNIRequestSigned`
  - `WantMNIResponseSigned`

5 Remove the hosted identity provider metadata by running the following command:

`saml2meta delete -u amadmin -w` *admin-password* `-e` *idp-entityID*

6 Import the new hosted identity provider metadata by running the following command:

`saml2meta import -u amadmin -w` *admin-password* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name* `-t` *COT-name*

7 Remove the remote service provider metadata by running the following command:

`saml2meta delete -u amadmin -w` *admin-password* `-e` *sp-entityID*

8 **Get the new remote service provider metadata.**

The instructions in this step assume a testing environment where you are in control of both the identity provider server and the service provider server.

a. **Copy** `spMeta.xml` **and** `spExtended.xml` **from the service provider machine.**

b. **Change hosted="1" to** `hosted="0"` **in** `spExtended.xml`**.**

9 **Import the new remote service provider metadata by running the following command:**

`saml2meta import -u amadmin -w` *admin-password* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name* `-t` *COT-name*

10 **Remove the remote identity provider metadata by running the following command:**

`saml2meta delete -u amadmin -w` *admin-password* `-e` *idp-entityID*

11 **Get the new remote identity provider metadata.**

The instructions in this step assume a testing environment where you are in control of both the identity provider server and the service provider server.

a. **Copy** `idpMeta.xml` **and** `idpExtended.xml` **from the identity provider machine.**

b. **Change hosted="1" to** `hosted="0"` **in** `idpExtended.xml`**.**

12 **Import the new remote identity provider metadata by running the following command:**

`saml2meta import -u amadmin -w` *admin-password* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name* `-t` *COT-name*

13 **Remove the hosted service provider metadata by running the following command:**

`saml2meta delete -u amadmin -w` *admin-password* `-e` *sp-entityID*

14 **Import the new hosted service provider metadata by running the following command:**

`saml2meta import -u amadmin -w` *admin-password* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name* `-t` *COT-name*

15 **Restart your web container.**

Securing SOAP Binding

SOAP binding supports the following authentication methods to protect interactions between SAML v2 entities:

- "Basic Authentication" on page 74
- "Secure Socket Layer/Transport Layer Security" on page 74

Basic Authentication

Once basic authentication is set up to protect a SAML v2 SOAP endpoint, all entities communicating with this endpoint must configure three basic authentication-related attributes in the extended metadata as described in the following table.

TABLE 4–1 Securing SOAP Endpoint with Basic Authentication

Attribute	Description
`basicAuthOn`	Establishes that the SOAP endpoint is using basic authentication. Takes a value of `true` or `false`.
`basicAuthUser`	Defines the user allowed access to the protected SOAP endpoint in the original SAML v2 entity.
`basicAuthPassword`	Defines an encrypted password for the user. The password is encrypted using `ampassword` on the partner side. For information on `ampassword`, see *Sun Java System Access Manager 7 2005Q4 Administration Guide*.

To modify the metadata, you must first export it to a file. Once you've modified the values of the applicable attributes, the metadata must be reloaded using the `saml2meta` command and the web container must be restarted. For more information, see "The `saml2meta` Command-line Reference" on page 57.

Secure Socket Layer/Transport Layer Security

Secure Socket Layer/Transport Layer Security (SSL/TLS) can also be enabled to protect SOAP endpoints and secure communications between SAML v2 entities. When one SAML v2 entity initiates communication with a SAML v2 entity deployed in an SSL/TLS-enabled web container, the initiating entity is referred to as the SSL/TLS client and the replying entity is referred to as the SSL/TLS server.

Server Certificate Authentication

For SSL/TLS server authentication (the server needs to present a certificate to the client), the following properties need to be set in the Virtual Machine for the Java^{TM} platform (JVM^{TM}) running the SSL/TLS client:

`-Djavax.net.ssl.trustStore`	Defines the full path to the file containing the server's CA certificate(s).
`-Djavax.net.ssl.trustStoreType`	Takes a value of `JKS` (Java Key Store).

In addition, the client's CA certificate needs to be imported into the certificate store/database of the server's web container and marked as a trusted issuer of client certificates.

Client Certificate Authentication

For SSL/TLS client authentication (the client needs to present a certificate to the server), the following properties need to be set in the JVM software running the SSL/TLS client:

`-Djavax.net.ssl.keyStore`	Defines the full path to the keystore containing the client certificate and private key. This may be the same as that defined in "Server Certificate Authentication" on page 75.
`-Djavax.net.ssl.keyStoreType`	Takes a value of `JKS`.
`-Djavax.net.ssl.keyStorePassword`	Specifies the password to the keystore.

On the SSL/TLS server side, the client's CA certificate needs to be imported into the web container's keystore and marked as a trusted issuer of client certificates.

Load Balancing

In cases of large deployments, a load balancer can be put in front of multiple instances of Access Manager and Federation Manager that have the SAML v2 Plug-in for Federation Services installed. The following procedure describes how to enable load balancer support.

▼ To Enable Load Balancer Support

1 **Install Access Manager and Federation Manager and follow the documentation to set up a load balancer.**

Load balancing information for Access Manager can be found in *Sun Java System Access Manager 7 2005Q4 Deployment Planning Guide*.

2. **Install the SAML v2 Plug-in for Federation Services on any machines acting as an identity provider or a service provider.**

3. **On any service provider machines, copy the metadata configuration files into the same directory and rename as follows:**
   - `spMeta.xml` to `spMeta.xml.lb`
   - `spExtended.xml` to `spExtended.xml.lb`

4. **Edit the new service provider load balancer metadata configuration files as follows:**
   - Change the host name of the service provider to that of the load balancer on the service provider side.
   - Change the port of the service provider to that of the load balancer on the service provider side.
   - Change the `metaAlias` of the service provider to any new `metaAlias`, for example, `/splb`.

5. **On any identity provider machines, copy the metadata configuration files into the same directory and rename as follows:**
   - `idpMeta.xml` to `idpMeta.xml.lb`
   - `idpExtended.xml` to `idpExtended.xml.lb`

6. **Edit the new identity provider load balancer metadata configuration files as follows:**
   - Change the host name of the identity provider to that of the load balancer on the identity provider side.
   - Change the port of the identity provider to that of the load balancer on the identity provider side.
   - Change the `metaAlias` of the identity provider to any new `metaAlias`, for example, `/idplb`.

7. **Import the new hosted metadata onto the service provider machines.**

8. **Import the new remote identity provider metadata onto the service provider machines.**

9. **Import the new hosted metadata onto the identity provider machines.**

10. **Import the new remote service provider metadata onto the identity provider machines.**

11. **Restart the web containers.**

Access Control

The following procedure will allow user access on the service provider side based on the user's configured roles on the identity provider side. This information is passed to the service provider in an assertion. No matching user entry is necessary on the service provider side.

▼ To Enable Access Control Using Agents and Roles

- You must configure the agent to enforce policy. For example, setting `com.sun.am.policy.agents.config.do_sso_only=false`.
- You must configure the attribute mapper on the identity provider side to include the required attributes in the `AttributeStatment`.
- You must configure the service provider to set the received attributes as key/value pairs in the user's single sign-on token. The agent will set those attributes in the HTTP header, passing them down to the application.

1 **Install the SAML v2 Plug-in for Federation Services on the identity provider.**

2 **Install the SAML v2 Plug-in for Federation Services on the service provider.**

Note – The service provider must be an instance of Access Manager because Federation Manager does not currently support policy.

---

3 **Install the Sun Java System Policy Agents 2.2 to protect the service provider configured on the instance of Access Manager.**

For more information, see the *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

4 **Modify** `com.sun.am.policy.am.login.url` **in** `AMAgent.properties` **so that its value is a URL (appended with the** `NameIDFormat=transient` **query parameter) that points to a single sign-on JSP on the service provider side.**

```
com.sun.am.policy.am.login.url=SP-protocol://SP-host:SP-port/service-deploy-uri/
saml2/jsp/spSSOInit.jsp?NameIDFormat=transient&metaAlias=SP-metaAlias&
idpEntityID=IDP_EntityID
```

For example:

```
com.sun.am.policy.am.login.url=http://moonriver.red.sun.com:58080/
amserver/saml2/jsp/spSSOInit.jsp?NameIDFormat=transient&metaAlias=/sp&
idpEntityID=sunriver.central.sun.com
```

5. **(Required only if using Web Agent 2.1) Set the value of the `com.sun.am.policy.am.library.loginURL` property to the service provider's login URL so the agent can authenticate itself.**

   If the login URL is a URL that initiates a SAML v2 single sign-on interaction, the value of this property will be used to authenticate the agent itself to your instances of Access Manager or Federation Manager. An example value might be **`http://`***host*:*port***`/amserver/UI/Login`**.

6. **Modify `spSSOInit.jsp` on the service provider side to use `goto` parameter as the value for `RelayState`.**

   The differences are as follows:

```
***************
*** 143,148 ****
--- 143,154 ----
}
idpEntityID = request.getParameter("idpEntityID");
paramsMap = SAML2Utils.getParamsMap(request);
+ String gotoURL = (String) request.getParameter("goto");
+ if (gotoURL != null) {
+ List list = new ArrayList();
+ list.add(gotoURL);
+ paramsMap.put(SAML2Constants.RELAY_STATE, list);
+ }
if ((idpEntityID == null) || (idpEntityID.length() == 0)) {
// get reader url
```

7. **Set up single sign-on without requiring writes to the data store by following the procedure described in "To Configure Single Sign-on Without Data Store Writes" on page 67.**

   To test, we assume the `employeenumber` attribute stores the user's role. In addition, the identity provider should have the following configured users:

   - User 1 has `employeenumber` set to `manager` (the manager's role).
   - User 2 has `employeenumber` set to `employee` (the employee's role).

8. **Create a policy with the `SessionProperty` condition on the service provider instance of Access Manager.**

   a. **Log in to the Access Manager console as the top-level administrator, by default, `amadmin`.**

   b. **Under the Access Control tab, select the appropriate realm.**

   c. **Select the Policies tab.**

   d. **Click New Policy.**

   e. **Enter a name for the policy.**

- f. **Click New under Rules.**
- g. **Select URL Policy Agent (with resource name) and click Next.**
- h. **Enter a name for the rule.**
- i. **Enter the application's URL as the value for Resource Name.**
- j. **Select Allow under both GET and POST and click Finish.**
- k. **Click New under Conditions.**
- l. **Select SessionProperty and click Next.**
- m. **Enter a name for the condition.**
- n. **Click Add under Values.**
- o. **Enter the single sign-on token property name as the value for Property Name.**

  To test, we will use `employeenumber`.
- p. **Add the match value to the Values field and click Add.**

  To test, we will use `manager`.
- q. **Click Add to return to the New Condition page.**
- r. **Click Finish to save the condition.**
- s. **Click Create to create the policy.**

For more information on creating policy, see the *Sun Java System Access Manager 7 2005Q4 Administration Guide*.

9 **Access the application using a web browser.**

You will be redirected to the service provider single sign-on JSP defined in the previous step. From there, you will be redirected to the identity provider to login. Single sign-on with the service provider will be accomplished using SAML v2 and, finally, you will be redirected back to the application for policy enforcement. If you logged in as User 1, you will be allowed to access the application as a manager which is allowed by the policy. If you logged in as User 2, an employee, you will be denied access to the application.

Certificate Revocation List Checking

The certificate revocation list (CRL) is a list of revoked certificates that contains the reason(s) for the certificate's revocation, the date of it's issuance, and the entity that issued it. When a potential user attempts to access the Access Manager or Federation Manager server, first access is allowed or denied based on the CRL entry for the root certificate included with the request. When the SAML v2 Service receives the incoming XML request, it parses the issuer Distinguished Name (DN) from the root certificate and retrieves the value defined by the `com.sun.identity.crl.cache.directory.searchattr` attribute in `AMConfig.properties`. If the attribute value is `CN` and the issuer DN is, for example, `CN="Entrust.net Client Certification Authority", OU=...`, the SAML v2 Service uses *Entrust.net Client Certification Authority* to retrieve the CRL from the LDAP directory which acts as the CRL repository.

Note – The LDAP directory which acts as the CRL repository is also configured in `AMConfig.properties`.

With this action, one of the following will occur:

1. If the LDAP directory returns a CRL that is not valid, the SAML v2 Service retrieves the value of the `IssuingDistributionPointExtension` attribute (usually an HTTP or LDAP URI) from the CRL and uses it to get new CRL from the certificate authority. If the certificate authority returns a valid CRL, it is saved to the LDAP directory and to memory and used for certificate validation.
2. If the LDAP directory returns no CRL but the certificate that is being validated has a defined `CRL Distribution Point Extension`, the SAML v2 Service retrieves it's value (usually an HTTP or LDAP URI) and uses the value to get a new CRL from the certificate authority. If the certificate authority returns a valid CRL, it is saved to the LDAP directory and to memory and used for certificate validation.
3. If the certificate authority returns a valid CRL, it is saved to the LDAP directory and to memory and used for certificate validation.

Note – Currently, Certificate Revocation List Checking works only with an instance of Sun Java System Directory Server.

After the CRL is loaded into memory and the root certificate validation is successful, the single sign-on process continues with validation of the signed XML message. The following are procedures to set up the SAML v2 Service for CRL checking.

- “To Set Up for Certificate Revocation List Checking” on page 81
- “To Manually Populate a Directory Server with a Certificate Revocation List” on page 83

Caution – CRL checking currently only works in the case of XML-based signature validation; for example, service provider side POST Artifact profile, or SOAP based logout. CRL checking does not work in the case of URL string based signature validation, XML signing, XML encryption or decryption.

▼ To Set Up for Certificate Revocation List Checking

Before You Begin A local instance of Directory Server must be designated as the CRL repository. It can be the same directory in which the Access Manager or Federation Manager schema is stored or it can be standalone. The Java Development Kit (JDK) must be version 1.5 or higher.

Note – If enabling this feature on an instance of Access Manager, it must be Access Manager version 7.0sp5 and above.

1 **Create one entry in Directory Server for each certificate authority.**

For example, if the certificate authority's `subjectDN` is `CN="Entrust.net Client Certification Authority",OU="www.entrust.net/GCCA_CPS incorp. by ref. (limits lib.)",O=Entrust.net` and the base DN for Directory Server is `dc=sun,dc=com`, create an entry with the DN `cn="Entrust.net Client Certification Authority",ou=people,dc=sun,dc=com`.

Note – If the certificate authority's `subjectDN` does not contain `uid` or `cn` attributes, do the following:

a. **Create a new object class.**

   For example, `sun-am-managed-ca-container`.

b. **Populate the new object class with the following attributes:**

   - `objectclass`
   - `ou`
   - `authorityRevocationList`
   - `caCertificate`
   - `certificateRevocationList`
   - `crossCertificatePair`

c. **Add the following entry (modified per your deployment) to Directory Server.**

```
dn: ou=1CA-AC1,dc=sun,dc=com
objectClass: top
objectClass: organizationalunit
objectClass: iplanet-am-managed-ca-container
ou: 1CA-AC1
```

You will publish the appropriate CRL to the entry created in the last step.

---

2 **Publish the appropriate CRL to the corresponding LDAP entry.**

This part can be done automatically by Access Manager or Federation Manager or manually. If the certificate being validated has a `CRL Distribution Point Extension` value, the publishing of the CRL is done automatically. If the certificate being validated has an `IssuingDistributionPointExtension` value, the initial publishing of the CRL must be done manually but future updates are done in runtime. If the certificate being validated has neither of these values, updates must be done manually at all time. See "To Manually Populate a Directory Server with a Certificate Revocation List" on page 83 for information on manual population.

3 **Configure the following properties in `AMConfig.properties` to point to the instance of Directory Server designated as the CRL repository.**

- `com.sun.identity.saml2.crl.cache.directory.host` defines the LDAP directory's host name.
- `com.sun.identity.saml2.crl.cache.directory.port` defines the LDAP directory's port number.
- `com.sun.identity.saml2.crl.cache.directory.ssl` takes a vale of TRUE or FALSE.
- `com.sun.identity.saml2.crl.cache.directory.user` defines the DN of the user with permission to bind to the LDAP directory.

- `com.sun.identity.saml2.crl.cache.directory.password` defines the encrypted password for the bind user. Use `ampassword` for the encryption. SeeChapter 2, "The ampassword Command Line Tool," in *Sun Java System Access Manager 7.1 Administration Reference* for more information.
- `com.sun.identity.saml2.crl.cache.directory.searchloc` defines the base DN from where the search will begin.
- `com.sun.identity.saml2.crl.cache.directory.searchattr` defines the component of the root certificate's `subjectDN` (issuer) that will be used to retrieve the CRL from LDAP directory. The value is a single string as in `cn`.

Note – All root certificate authorities must use the same search attribute.

---

com.sun.identity.saml2.crl.cache.directory.password defines the password for the bind user. This actually need to be the encrypted password of the bind user, customer need to use ampassword to encrypt the password before putting values here.

4 **Import all the certificate authority certificates into the `cacerts` keystore under the `java.home/jre/lib/secure` directory using the `keytool` utility.**

Certificates must be imported as `trustedcacert`. More information on `keytool` can be found at `http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html`.

▼ To Manually Populate a Directory Server with a Certificate Revocation List

1 **Use your browser to get the initial CRL from the certificate authority manually.**

2 **Save the initial CRL file in the binary DER format to your local machine.**

3 **Convert the DER file to the text-based PEM format and finally LDAP Data Interchange Format (LDIF) using the following command:**

`ldif -b` *certificaterevocationlist;binary* **`< famouseCA.crl > crl.ldif`**

Note – The `ldif` command is available in your Directory Server installation.

The `crl.ldif` file contains text similar to the following:

```
certificaterevocationlist;binary:: MIH7MIGmMA0GCSqGSIb3DQEBBQUAMGExCzAJBgNVBA
  YTAlVTMRgwFgYDVQQKEw9VLlMuIEdvdmVybm1lbnQxDDAKBgNVBAsTA0RvRDEMMAoGA1UECxMDUE
  tJMRwwGgYDVQQDExNEb0QgQ2xhc3MgMyBSb290IENBFw0wNzA1MDExNDMzMDNaFw0wNzA1MDExNz
```

```
UzMDNaMBQwEgIBTxcNMDcwNDI3MTY1NzMzWjANBgkqhkiG9w0BAQUFAANBADUd7lBe7JeQKQnKCK
GddnsCXqii7EitbPuYT55M4Nn3qBgPFSG8bX9H5XBGTB4iofb3h0Y9DCqe10vc8dBM0
```

4 **Do one of the following to define the LDAP entry in which the CRL will be stored.**

- **For an existing entry, specify the DN in the LDIF file.**

```
# entry-id: famouseCA dn: CN=famouseCA,ou=People,dc=sun,dc=com
certificaterevocationlist;binary:: MIH7MIGmMA0GCSqGSIb3DQEBBQUAMGExCzAJBgNVBA
  YTAlVTMRgwFgYDVQQKEw9VLlMuIEdvdmVybm1lbnQxDDAKBgNVBAsTA0RvRDEMMAoGA1UECxMDUE
  tJMRwwGgYDVQQDExNEb0QgQ2xhc3MgMyBSb290IENBFw0wNzA1MDExNDMzMDNaFw0wNzA1MDExNz
  UzMDNaMBQwEgIBTxcNMDcwNDI3MTY1NzMzWjANBgkqhkiG9w0BAQUFAANBADUd7lBe7JeQKQnKCK
  GddnsCXqii7EitbPuYT55M4Nn3qBgPFSG8bX9H5XBGTB4iofb3h0Y9DCqe10vc8dBM0
```

- **For a new entry, specify the DN and object classes in the LDIF file.**

```
# entry-id: tester200
dn: CN=famouseCA,ou=People,dc=sun,dc=com
sn: famouseCA
cn: famouseCA
employeeNumber: 1001
telephoneNumber: 555-555-5555
postalAddress: 555 Test Drive
iplanet-am-modifiable-by: cn=Top-level Admin Role,dc=iplanet,dc=com
mail: famouseCA@test.com
givenName: Test
inetUserStatus: Active
uid: tester200
objectClass: iplanet-am-user-service
objectClass: inetAdmin
objectClass: iPlanetPreferences
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: iplanet-am-managed-person
objectClass: inetuser
objectClass: top
userPassword: {SSHA}E3TJ4DT7IoOLETVny1ktxUGWNTpBYq8tj3C1Sg==
creatorsName: cn=puser,ou=dsame users,dc=iplanet,dc=com
modifiersName: cn=puser,ou=dsame users,dc=iplanet,dc=com
createTimestamp: 20031125043253Z
modifyTimestamp: 20031125043253Z
certificaterevocationlist;binary:: MIH7MIGmMA0GCSqGSIb3DQEBBQUAMGExCzAJBgNVBA
  YTAlVTMRgwFgYDVQQKEw9VLlMuIEdvdmVybm1lbnQxDDAKBgNVBAsTA0RvRDEMMAoGA1UECxMDUE
  tJMRwwGgYDVQQDExNEb0QgQ2xhc3MgMyBSb290IENBFw0wNzA1MDExNDMzMDNaFw0wNzA1MDExNz
  UzMDNaMBQwEgIBTxcNMDcwNDI3MTY1NzMzWjANBgkqhkiG9w0BAQUFAANBADUd7lBe7JeQKQnKCK
  GddnsCXqii7EitbPuYT55M4Nn3qBgPFSG8bX9H5XBGTB4iofb3h0Y9DCqe10vc8dBM0G8=
```

5 Run one of the following `ldapmodify` commands based on whether you are adding the LDIF file to an existing entry or creating a new entry.

- **To add a CRL to an existing LDAP entry (using an LDIF file with a specified DN), use the following command:**

  ```
  ldapmodify -r -h Directory Server_host -p Directory Server_port
  -f ldif-file -D cn=Directory Manager -w password
  ```

- **To add a CRL to a new LDAP entry (using an LDIF file with a specified DN and object classes), use the following command:**

  ```
  ldapmodify -a -h Directory Server_host -p Directory Server_port
  -f ldif-file -D cn=Directory Manager -w password
  ```

Bootstrapping the Liberty ID-WSF with SAML v2

SAML v2 can be used to bootstrap into the Liberty Alliance Project Identity Web Services Framework (Liberty ID-WSF) version 1.1. For example, a service provider communicating with the SAML v2 specifications might want to communicate with web services based on the Liberty ID-WSF regarding a principal. To do this, the SAML v2 Assertion returned to the service provider must contain a Discovery Service endpoint. The service provider than acts as a web services consumer, using the value included within the Endpoint tag to bootstrap the Discovery Service. This then allows access to other Liberty ID-WSF services.

A sample SAML v2 assertion is reproduced below. Note the SAML v2 security token stored in the Discovery Service resource offering: `urn:liberty:security:2003-08:null:SAML`. Both are stored within the attribute statement.

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Version="2.0"
ID="s21bdfd298f332ef2ada1d4fd00bab21c0f64cc90a"
IssueInstant="2007-03-27T08:25:26Z">
<saml:Issuer>http://hengming.red.iplanet.com</saml:Issuer>
<saml:Subject>
<saml:NameID NameQualifier="http://hengming.red.iplanet.com"
  SPNameQualifier="http://isdev-2.red.iplanet.com" Format=
  "urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
  HuCJIy9v5MdrjJQOgsuT4NWmVUl3</saml:NameID>
<saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
<saml:SubjectConfirmationData NotOnOrAfter="2007-03-27T08:35:26Z"
  InResponseTo="s20711ed113989a9bff544f61c700d0bd0a08b78fd"
  Recipient="http://isdev-2.red.iplanet.com:58080/
  amserver/Consumer/metaAlias/sp"  >
  </saml:SubjectConfirmationData>
  </saml:SubjectConfirmation>
  </saml:Subject>
<saml:Conditions NotBefore="2007-03-27T08:25:26Z"
```

```
  NotOnOrAfter="2007-03-27T08:35:26Z">
<saml:AudienceRestriction>
<saml:Audience>http://isdev-2.red.iplanet.com</saml:Audience>
  </saml:AudienceRestriction>
  </saml:Conditions>
<saml:AuthnStatement AuthnInstant="2007-03-27T08:19:24Z"
  SessionIndex="s234f01958bf364aff26829d9d9846ba51afc2b201">
  <saml:AuthnContext>
  <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:
  2.0:ac:classes:PasswordProtectedTransport
  </saml:AuthnContextClassRef>
  </saml:AuthnContext>
  </saml:AuthnStatement>
<saml:AttributeStatement>
<saml:Attribute Name="offerings" NameFormat="urn:liberty:disco:2003-08">
<saml:AttributeValue xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
<ResourceOffering xmlns="urn:liberty:disco:2003-08">
<ResourceID xmlns="urn:liberty:disco:2003-08">http://hengming.red.iplanet.com
  /aWQ9aWRwLG91PXVzZXIsZGM9aXBsYW5ldCxkYz1jb20sYW1zZGtkbj11aWQ9aWRwLG91PXBlb3BsZ
  SxkYz1pcGxhbmV0LGRjPWNvbQ%3D%3D</ResourceID>
<ServiceInstance xmlns="urn:liberty:disco:2003-08">
<ServiceType>urn:liberty:disco:2003-08</ServiceType>
<ProviderID>http://hengming.red.iplanet.com</ProviderID>
<Description xmlns="urn:liberty:disco:2003-08"
  id="sf6a6d3dcc16e729eea0d7e5587a5ff27f234f991">
<SecurityMechID>urn:liberty:security:2003-08:null:SAML
  </SecurityMechID>
<CredentialRef>s5dc88819de075e4e9c8db3deb8b46d4d8758b4b901
  </CredentialRef>
<Endpoint>http://hengming.red.iplanet.com:58080/amserver/Liberty/disco
  </Endpoint></Description>
  </ServiceInstance></ResourceOffering></saml:AttributeValue></saml:Attribute>
<saml:Attribute Name="credentials" NameFormat="urn:liberty:disco:2003-08">
<saml:AttributeValue xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
<saml:Assertion  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  MajorVersion="1" MinorVersion="1"
  AssertionID="s5dc88819de075e4e9c8db3deb8b46d4d8758b4b901" Issuer=
  "http://hengming.red.iplanet.com" IssueInstant="2007-03-27T08:25:26Z" >
<sec:ResourceAccessStatement xmlns:sec="urn:liberty:sec:2003-08">
<saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
<saml:NameIdentifier NameQualifier="http://isdev-2.red.iplanet.com"
  Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
  HuCJIy9v5MdrjJQOgsuT4NWmVUl3</saml:NameIdentifier>
<saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:sendervouches
  </saml:ConfirmationMethod>
  </saml:SubjectConfirmation>
  </saml:Subject>
```

```
<ResourceID xmlns="urn:liberty:disco:2003-08">http://hengming.red.iplanet.com/
aWQ9aWRwLG91PXVzZXIsZGM9aXBsYW5ldCxkYz1jb20sYW1zZG
tkbj11aWQ9aWRwLG91PXBlb3BsZSxkYz1pcGxhbmV
0LGRjPWNvbQ%3D%3D</ResourceID>
<sec:ProxySubject xmlns:sec="urn:liberty:sec:2003-08">
<saml:NameIdentifier xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
Format="urn:liberty:iff:nameid:entityID">http://isdev-2.red.iplanet.com
  </saml:NameIdentifier>
<saml:SubjectConfirmation xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#"><KeyName>CN=sun-unix, OU=SUN Java System
  Access Manager, O=Sun, C=US</KeyName><KeyValue><RSAKeyValue><Modulus>AOA/2kpfKFWvRXOMbrmTlKe102ibw/
  aTd3HBVgI8cHsywww8M1J0X+vJvvk6eabTNWY5jBfTo9i1bC4AXXoRlxgsE/
  6Uq5+6NGrd+iwfvj25x8HzHX8LrJ+7EzlGVsKO
  M+A3vTP0tCkmYE4jatZbWlRoto0wyInP2wMFdKPrmYWL</Modulus>
<Exponent>AQAB</Exponent></RSAKeyValue>
  </KeyValue></KeyInfo></saml:SubjectConfirmation>
  </sec:ProxySubject><sec:SessionContext xmlns:sec="urn:liberty:sec:2003-08" AuthenticationInstant=
  "2007-03-27T08:25:26Z" AssertionIssueInstant="2007-03-27T08:25:26Z">
<sec:SessionSubject xmlns:sec="urn:liberty:sec:2003-08">
<saml:NameIdentifier xmlns:saml="urn:oasis:names:tc:SAML:1.0:
  assertion" NameQualifier="http://isdev-2.red.iplanet.com"
  Format="urn:oasis:names:tc:SAML:
  2.0:nameid-format:persistent">HuCJIy9v5MdrjJQOgsuT4NWmVUl3
  </saml:NameIdentifier>
<saml:SubjectConfirmation xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:2.0:cm:bearer</saml:ConfirmationMethod>
</saml:SubjectConfirmation>
<lib:IDPProvidedNameIdentifier  xmlns:lib="http://projectliberty.org/
  schemas/core/2002/12"
  NameQualifier="http://hengming.red.iplanet.com" Format="urn:oasis:names:tc:SAML:2.0:
  nameid-format:persistent"  >HuCJIy9v5MdrjJQOgsuT4NWmVUl3
  </lib:IDPProvidedNameIdentifier>
  </sec:SessionSubject>
<sec:ProviderID>http://hengming.red.iplanet.com</sec:ProviderID>
  <lib:AuthnContext xmlns:lib="urn:liberty:iff:2003-08"><lib:AuthnContextClassRef>urn:oasis:
  names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</lib:AuthnContextClassRef>
  <lib:AuthnContextStatementRef>http://www.projectliberty.org/schemas/authctx/classes/
  Password</lib:AuthnContextStatementRef></lib:AuthnContext></sec:SessionContext>
  </sec:ResourceAccessStatement>
<saml:AuthenticationStatement xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
  AuthenticationInstant="2007-03-27T08:19:24Z">
<saml:Subject>
<saml:NameIdentifier Format="urn:liberty:iff:nameid:entityID">
  http://isdev-2.red.iplanet.com</saml:NameIdentifier>
<saml:SubjectConfirmation>
```

```
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#"><KeyName>CN=sun-unix, OU=SUN Java System
  Access Manager, O=Sun, C=US</KeyName><KeyValue><RSAKeyValue><Modulus>AOA/2kpfKFWvRXOMbrmTlKe102ibw/
  aTd3HBVgI8cHsywww8M1J0X+vJvvk6eabTNWY5jBfTo9i1bC4AXXoRlxgsE/6Uq
  5+6NGrd+iwfvj25x8HzHX8LrJ+7EzlGVsKOM+
  A3vTP0tCkmYE4jatZbWlRoto0wyInP2wMFdKPrmYWL</Modulus>
<Exponent>AQAB</Exponent>
  </RSAKeyValue>
  </KeyValue></KeyInfo></saml:SubjectConfirmation>
</saml:Subject>
</saml:AuthenticationStatement>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="#s5dc88819de075e4e9c8db3deb8b46d4d8758b4b901">
<Transforms>
<Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>td1CqmbWC5eMXCK6IFhzZxn3GJg=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>
YJ4g+jV5KIQRpkI9jlsZMbKx9lBhEB5ngB8NrH5nPh8+XFTK2gPZNzovOYOzxlznuxxbvC3A4rpg
UoSeE3N+oE4sl5KnY1GewFgjckAdeWafcLhGd9O68A+9nqMnRW/5fR9mnbk9eqZO8zx2bO8toiWi
pQCTU5XcDYkCNb8LgFs=
</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>
MIICOTCCAeOgAwIBAgIBEjANBgkqhkiG9w0BAQQFADBFMQswCQYDVQQGEwJVUzEYMBYGA1UEChMP
cmVkLmlwbGFuZXQuY29tMRwwGgYDVQQDExNDZXJ0aWZpY2F0ZSBNYW5hZ2VyMB4XDTA1MDYwMjE3
NTkxOFoXDTA2MDYwMjE3NTkxOFowVzELMAkGA1UEBhMCVVMxDDAKBgNVBAoTA1N1bjEnMCUGA1UE
CxMeU1VOIEphdmEgU3lzdGVtIEFjY2VzcyBNYW5hZ2VyMREwDwYDVQQDEwhzdW4tdW5peDCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA4D/aSl8oVa9Fc4xuuZOUp7XTaJvD9pN3ccFWAjxwezLD
DDwzUnRf68m++Tp5ptM1ZjmMF9Oj2LVsLgBdehGXGCwT/pSrn7o0at36LB++PbnHwfMdfwusn7sT
OUZWwo4z4De9M/S0KSZgTiNq1ltaVGi2jTDIic/bAwV0o+uZhYsCAwEAAaNoMGYwEQYJYIZIAYb4
QgEBBAQDAgZAMA4GA1UdDwEB/wQEAwIE8DAfBgNVHSMEGDAWgBQqOASyzZ41LergF+cSQN1Gokpa
XjAgBgNVHREEGTAXgRVoZW5nLW1pbmcuaHN1QHN1bi5jb20wDQYJKoZIhvcNAQEEBQADQQDKxdPy
821aQRVZ0wLqa6LBYZCUcZD5AMvzl3EylwtniHmzPtOeZe4NmFj7qQziSb1H57NSkiwKaLZ7Mt6F
jaUU
</X509Certificate>
</X509Data>
</KeyInfo>
</Signature></saml:Assertion>
</saml:AttributeValue></saml:Attribute></saml:AttributeStatement></saml:Assertion>
```

Following are the procedures to enable bootstrapping of the Liberty ID-WSF Discovery Service using SAML v2.

- "To Enable an Identity Provider for SAML v2 Bootstrapping of Liberty ID-WSF" on page 89
- "To Enable a Service Provider for SAML v2 Bootstrapping of Liberty ID-WSF" on page 90

▼ To Enable an Identity Provider for SAML v2 Bootstrapping of Liberty ID-WSF

Before You Begin See "The `saml2meta` Command-line Reference" on page 57 for more information on the command line interface used in this procedure.

1 Choose one of the following options to get metadata for the appropriate identity provider.

The option you choose is dependent on where you are in the process of configuring the identity provider.

- **If metadata for the identity provider you are configuring has not yet been imported, or signing and encryption certificate aliases have not been configured in the existing identity provider metadata, generate standard and extended metadata templates for the identity provider using the `saml2meta` command line interface.**

  ```
  saml2meta template -u amadmin -w amadmin_pw -d /idp
  -b certificate_alias -g enc_certificate_alias -e http://host_machine
  -m standard_meta_filename -x extended_meta_filename
  ```

- **If the identity provider metadata has been imported, and signing and encryption keys have all been configured, export the existing extended entity configuration metadata of the identity provider using the `saml2meta` command line interface.**

  ```
  saml2meta export -u amadmin -w amadmin_pw -d /idp
   -e http://host_machine -x extended_meta_filename
  ```

2 Edit the identity provider's extended entity configuration template by changing the value of `discoveryBootstrappingEnabled` to `true`.

The extended entity configuration template is *extended_meta_filename* created in the previous step. If the attribute doesn't exist in the metadata, add the following lines above the ending tag `</IDPSSOConfig>`.

```
<Attribute name="discoveryBootstrappingEnabled">
<Value>true</Value>
</Attribute>
```

3 **(Optional) Delete the current metadata for the identity provider using the `saml2meta` command line interface.**

The option you choose in this step is dependent on the option chosen in the first step of this procedure.

- **If you choose the first option in this procedure's first step, delete the current standard and extended metadata using the `saml2meta` command line interface.**

  ```
  saml2meta delete -u amadmin -w amadmin_pw -e http://host_machine
  ```

- **If you choose the second option in this procedure's first step, delete the current extended metadata only using the `saml2meta` command line interface.**

  ```
  saml2meta delete -u amadmin -w amadmin_pw -e http://host_machine -c
  ```

4 **Import the new identity provider metadata.**

The option you choose in this step is dependent on the option chosen in the first step of this procedure. *circle_of_trust* is the name of the circle of trust into which you are importing these files.

- **If you choose the first option in this procedure's first step, import the standard metadata and the modified extended metadata files using the `saml2meta` command line interface.**

  ```
  saml2meta import -u amadmin -w amadmin_pw
  -m standard_meta_filename -x extended_meta_filename -t circle_of_trust
  ```

- **If you choose the second option in this procedure's first step, import the modified extended metadata using the `saml2meta` command line interface.**

  ```
  saml2meta import -u amadmin -w amadmin_pw
   -x extended_meta_filename -t circle_of_trust
  ```

5 **Add the following line to the end of the `AMConfig.properties` file to enable Liberty ID-WSF to work with SAML v2 on the identity provider.**

```
com.sun.identity.liberty.ws.uti.providerManagerClass=com.sun.identity.saml2.plugins.SAML2P
```

6 **Restart the web container.**

▼ To Enable a Service Provider for SAML v2 Bootstrapping of Liberty ID-WSF

1 **Add the following line to the end of the `AMConfig.properties` file to enable Liberty ID-WSF to work with SAML v2 on the identity provider.**

```
com.sun.identity.liberty.ws.uti.providerManagerClass=com.sun.identity.saml2.plugins.SAML2P
```

2 **(Optional) Add the following to the class path of the web application.**

```
/opt/SUNWam/saml2/lib/saml2.jar
```

This step is necessary only if the web application you are protecting is using the same Java Virtual Machine (JVM) as the instance of Access Manager or Federation Manager. In this situation, you can use the following API to retrieve the Discovery Service bootstrap resource offering and included security token

```
ResourceOffering SAML2SDKUtil.getDiscoveryBootStrapResourceOffering(
       HttpServletRequest request)
List SAML2SDKUtil.getDiscoveryBootStrapCredentials(
       HttpServletRequest request)
```

3 **Restart the web container.**

CHAPTER 5

Developer Tools

The SAML v2 Plug-in for Federation Services can be installed in an instance of Access Manager or Federation Manager, and integrated with your application to implement the SAML v2 profiles. After installation, included service provider interfaces (SPI) and JavaServer Pages™ (JSP™) can be used for customization. Alternately, the SAML v2 Plug-in for Federation Services software development kit (SDK) itself can be deployed inside your application to implement proprietary SAML v2 profiles.

This chapter covers the following topics:

- "The SAML v2 Plug-in for Federation Services SDK" on page 93
- "Service Provider Interfaces" on page 97
- "JavaServer Pages" on page 103

The SAML v2 Plug-in for Federation Services SDK

The SAML v2 Plug-in for Federation Services provides application programming interfaces (API) that can be used to construct and process assertions, requests, and responses. The SAML v2 Plug-in for Federation Services SDK is designed to be pluggable although it can also be run as a standalone application (outside of an instance of Access Manager or Federation Manager).

- For information on the packages in the SDK, see "The SDK Packages" on page 93.
- For ways to set a customized implementation, see "Setting a Customized Class" on page 94.
- For instructions on how to install the SDK as a standalone application, see "To Install the SAML v2 Plug-in for Federation Services SDK" on page 95.

The SDK Packages

The SAML v2 Plug-in for Federation Services SDK includes the following packages:

- "`com.sun.identity.saml2.assertion` Package" on page 94
- "`com.sun.identity.saml2.common` Package" on page 94

- "`com.sun.identity.saml2.protocol` Package" on page 94

For more information, see the *Sun Java System SAMLv2 Plug-in for Federation Services Java API Reference*.

Note – You can also extract and view the SAML v2 Plug-in for Federation Services Java API Reference internally. Change to the /*AccessManager-base*/*product-directory*/`saml2/docs` directory on Access Manager or the /*FederationManager-base*/`SUNWam/saml2/docs` directory on Federation Manager and extract `saml2_public_javadocs.jar` to your web container. The Java API Reference can then be viewed using a web browser.

`com.sun.identity.saml2.assertion` Package

This package provides interfaces to construct and process SAML v2 assertions. It also contains the `AssertionFactory`, a factory class used to obtain instances of the objects defined in the assertion schema.

`com.sun.identity.saml2.common` Package

This package provides interfaces and classes used to define common SAML v2 utilities and constants.

`com.sun.identity.saml2.protocol` Package

This package provides interfaces used to construct and process the SAML v2 request/response protocol. It also contains the `ProtocolFactory`, a factory class used to obtain object instances for concrete elements in the protocol schema.

Setting a Customized Class

There are two ways you could set a customized implementation class:

1. Add a mapping property to `AMConfig.properties` in the format:

   `com.sun.identity.saml2.sdk.mapping.`*interface-name*=*new-class-name*

   For example, to define a customized Assertion interface, you would add:

   ```
   com.sun.identity.saml2.sdk.mapping.Assertion=
    com.ourcompany.saml2.AssertionImpl
   ```

Note – `AMConfig.properties` is located in the `/etc/opt/`*product-directory*`/config` directory in Access Manager and in the `/`*staging-directory*`/web-src/WEB-INF/classes` directory in Federation Manager.

---

2. Set an environment variable for the Virtual Machine for the Java™ platform (JVM™). For example, you can add the following environment variable when starting the application:

```
-Dcom.sun.identity.saml2.sdk.mapping.Assertion=
 com.ourcompany.saml2.AssertionImpl
```

▼ To Install the SAML v2 Plug-in for Federation Services SDK

Before You Begin If installing the SDK on a Linux system, you must have the Red Hat Package Manager (RPM) installed.

1 Log in as root.

2 Create a new directory.

```
# mkdir saml2bits

# cd saml2bits
```

3 Download the ***file-name*****`.tar.gz` file into the new directory.**

See the *Sun Java System SAML v2 Plug-in for Federation Services Release Notes* for the download URL.

4 Unpack the product binaries by typing:

```
# gunzip –dc file-name.tar.gz | tar -xvof -
```

where *file-name*`.tar.gz` is the name of the downloaded file.

5 Add the SAML v2 packages as follows:

```
# pkgadd -d . SUNWsaml2
```

By default, the packages will be installed in `/`*AccessManager-base*`/`*product-directory*`/saml2` or `/`*FederationManager-base*`/SUNWam/saml2`.

6 Add the following to the classpath of your application:

- **For Access Manager 7 2005Q4:**

    - `/opt/`*product-directory*`/saml2/lib/saml2.jar`
    - `/opt/`*product-directory*`/saml2/locale`

- **For Federation Manager 7 2005Q4:**
    - `/opt/SUNWam/saml2/lib/saml2.jar`
    - `/opt/SUNWam/saml2/locale`

7 **Get the supporting JAR and locale files using the applicable procedure:**

- **For Access Manager 7 2005Q4:**

    a. **cd /***AccessManager-base***/***product-directory*

    b. **Run the following command:**

    ```
    # make -f Makefile.clientsdk
    ```

    c. **Add the following to the classpath of your application:**

    - *AccessManager-base***/***product-directory***`/clientsdk-webapps/WEB-INF/lib/amclientsdk.jar`**
    - *AccessManager-base***/***product-directory***`/clientsdk-webapps/WEB-INF/classes`**

- **For Federation Manager 7 2005Q4:**

    Add the following to the classpath of your application:

    a. *FederationManager-base*`/SUNWam/fm/web-src/WEB-INF/lib/am_services.jar`

    b. *FederationManager-base*`/SUNWam/fm/web-src/WEB-INF/lib/am_sdk.jar`

    c. *FederationManager-base*`/SUNWam/fm/web-src/WEB-INF/classes`

8 **Restart your application.**

You should now be able to process SAML v2 XML messages using the methods in the `AssertionFactory` and `ProtocolFactory`.

Next Steps For details regarding the SAML v2 SDK classes, see the *Sun Java System SAMLv2 Plug-in for Federation Services Java API Reference*.

Note – You can also extract and view the SAML v2 Plug-in for Federation Services Java API Reference internally. Change to the `/`*AccessManager-base*`/`*product-directory*`/saml2/docs` directory on Access Manager or the `/`*FederationManager-base*`/SUNWam/saml2/docs` directory on Federation Manager and extract `saml2_public_javadocs.jar` to your web container. The Java API Reference can then be viewed using a web browser.

Service Provider Interfaces

The `com.sun.identity.saml2.plugins` package provides pluggable interfaces to implement SAML v2 functionality into your application. The classes can be configured per provider entity. Default implementations are provided, but a customized implementation can be plugged in by modifying the corresponding attribute in the provider's extended metadata configuration file. The mappers include:

- "Account Mappers" on page 97
- "Attribute Mappers" on page 98
- "Authentication Context Mappers" on page 99

For more information, see the *Sun Java System SAMLv2 Plug-in for Federation Services Java API Reference*.

Note – You can also extract and view the SAML v2 Plug-in for Federation Services Java API Reference internally. Change to the `/`*AccessManager-base*`/`*product-directory*`/saml2/docs` directory on Access Manager or the `/`*FederationManager-base*`/SUNWam/saml2/docs` directory on Federation Manager and extract `saml2_public_javadocs.jar` to your web container. The Java API Reference can then be viewed using a web browser.

Account Mappers

An account mapper is used to associate a local user account with a remote user account based on a specified attribute. A default account mapper has been developed for both sides of the SAML v2 interaction, service providers and identity providers.

`IDPAccountMapper`

The `IDPAccountMapper` interface is used on the identity provider side to map user accounts in cases of single sign-on and federation termination. The default implementation, `com.sun.identity.saml2.plugins.DefaultIDPAccountMapper`, maps the accounts based on the persistent `NameID` attribute.

SPAccountMapper

The `SPAccountMapper` interface is used on the service provider side to map user accounts in cases of single sign-on and federation termination. The default implementation, `com.sun.identity.saml2.plugins.DefaultSPAccountMapper`, supports mapping based on the transient and persistent `NameID` attributes, and attribute federation based on properties defined in the extended metadata configuration file. The user mapping is based on information passed from the identity provider in an `<AttributeStatment>`.

Attribute Mappers

An attribute mapper is used to associate attribute names passed in the `<AttributeStatement>` of an assertion. A default attribute mapper has been developed for both participants in the SAML v2 interaction, service providers and identity providers. They are defined in the extended metadata configuration files and explained in the following sections:

- "`IDPAttributeMapper`" on page 98
- "`SPAttributeMapper`" on page 98
- "To Set Up Attribute Mappers" on page 99

IDPAttributeMapper

The `IDPAttributeMapper` interface is used by the identity provider to specify which user attributes will be included in an assertion. The default implementation, `com.sun.identity.saml2.plugins.DefaultIDPAttributeMapper`, retrieves attribute mappings (*SAML v2-attribute=user-attribute*) defined in the `attributeMap` property in the identity provider's extended metadata configuration file. It reads the value of the user attribute from the identity provider's data store, and sets this value as the `<AttributeValue>` of the specified SAML v2 attribute. The SAML v2 attributes and values are then included in the `<AttributeStatement>` of the assertion and sent to the service provider. The value of `attributeMap` can be changed to modify the mapper's behavior without programming. The default mapper itself can be modified to attach any identity provider user attribute with additional programming.

SPAttributeMapper

The `SPAttributeMapper` interface is used by the service provider to map attributes received in an assertion to its local attributes. The default implementation, `com.sun.identity.saml2.plugins.DefaultSPAttributeMapper`, retrieves the attribute mappings defined in the `attributeMap` property in the service provider's extended metadata configuration file. It extracts the value of the SAML v2 attribute from the assertion and returns a key/value mapping which will be set in the user's single sign-on token. The mapper can also be customized to choose user attributes from the local service provider datastore.

▼ To Set Up Attribute Mappers

This procedure will pass the `mail` and `employeeNumber` attributes from the identity provider to the service provider.

1. **Export the identity provider's current extended metadata configuration to a file.**

   `saml2meta [-i` *staging-directory*] **`export -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-x`** *IDP-extended-XML-file-name*

2. **Edit the `attributeMap` attribute in the exported extended metadata configuration file to include the user attributes the identity provider will pass to the service provider.**

   `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute-value*=*remote-provider-attribute*. For example,

   ```
   <Attribute name="attributeMap">
   <Value>mail=mail</Value>
   <Value>employeeNumber=employeeNumber</Value>
   </Attribute>
   ```

3. **Remove the identity provider's current extended metadata configuration.**

   `saml2meta [-i` *staging-directory*] **`delete -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-c`**

4. **Import the identity provider's modified extended metadata configuration file.**

   `saml2meta [-i` *staging-directory*] **`import -u amadmin -w`** *password* **`-x`** *IDP-extended-XML-file-name*

5. **Restart the web container.**

6. **Repeat the above steps for the service provider's extended metadata configuration file.**

7. **To test, invoke single sign-on from the service provider.**

   The assertion contains an `AttributeStatement` with the `mail` and `employeeNumber` attributes which will be set in the single sign-on token.

Authentication Context Mappers

Authentication context refers to information added to an assertion regarding details of the technology used for the actual authentication action. For example, a service provider can request that an identity provider comply with a specific authentication method by identifying that method in an authentication request. The authentication context mappers pair a standard SAMLv2 authentication context class reference (`PasswordProtectedTransport`, for example) to an Access Manager authentication scheme (`module=LDAP`, for example) on the identity provider side and set the appropriate authentication level in the user's SSO token on the service provider side. The identity provider would then deliver (with the assertion) the authentication context information in the form of an authentication context declaration added to the assertion. The process for this is described below.

1. A user accesses `spSSOInit.jsp` using the `AuthnContextClassRef` query parameter.

   For example, **`http://`***SP_host*`:`*SP_port*`/`*uri*`/`**`spSSOInit.jsp?`** **`metaAlias=`***SP_MetaAlias***`&idpEntityID=`***IDP_EntityID***`&AuthnContextClassRef=PasswordProtectedTr`**
2. The `SPAuthnContextMapper` is invoked to map the value of the query parameter to a `<RequestedAuthnContext>` and an authentication level.
3. The service provider sends the `<AuthRequest>` with the `<RequestedAuthnContext>` to the identity provider.
4. The identity provider processes the `<AuthRequest>` by invoking the `IDPAuthnContextMapper` to map the incoming information to a defined authentication scheme.

   Note – If there is no matching authentication scheme, an authentication error page is displayed.

   ---

5. The identity provider then redirects the user (including information regarding the authentication scheme) to the Authentication Service for authentication.

   For example, **`http://`***AM_host*`:`*AM_port*`/`*uri*`/`**`UI/Login?module=LDAP`** redirects to the LDAP authentication module.
6. After successful authentication, the user is redirected back to the identity provider for construction of a response based on the mapped authentication class reference.
7. The identity provider then returns the user to the assertion consumer on the service provider side.
8. After validating the response, the service provider creates a single sign-on token carrying the authentication level defined in the previous step.

A default authentication context mapper has been developed for both sides of the SAML v2 interaction. Details about the mappers are in the following sections:

- “`IDPAuthnContextMapper`” on page 100
- “`SPAuthnContextMapper`” on page 101

The procedure for configuring mappings is in the following section:

- “To Configure Mappings” on page 102

`IDPAuthnContextMapper`

The `IDPAuthnContextMapper` is configured for the identity provider and maps incoming authentication requests from the service provider to an Access Manager authentication scheme (user, role, module, level or service-based authentication), returning a response containing the authentication status to the service provider. The following attributes in the identity provider extended metadata are used by the `IDPAuthnContextMapper`:

- The `idpAuthncontextMapper` property specifies the mapper implementation.
- The `idpAuthncontextClassrefMapping` property specifies the mapping between a standard SAMLv2 authentication context class reference and an Access Manager authentication scheme. It takes a value in the following format:

  ```
  authnContextClassRef | authnType=authnValue | authnType=anthnValue | ...
  ```

  For example, `urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport|module=LDAP` maps the SAMLv2 `PasswordProtectedTransport` class reference to the Access Manager `LDAP` authentication module.

`SPAuthnContextMapper`

The `SPAuthnContextMapper` is configured for the service provider and maps the parameters in incoming HTTP requests to an authentication context. It creates a `<RequestedAuthnContext>` element based on the query parameters and attributes configured in the extended metadata of the service provider. The `<RequestedAuthnContext>` element is then included in the `<AuthnRequest>` element sent from the service provider to the identity provider for authentication. The `SPAuthnContextMapper` also maps the authentication context on the identity provider side to the authentication level set as a property of the user's single sign-on token. The following sections describe the parameters and attributes:

- "`SPAuthnContextMapper` Parameters" on page 101
- "`SPAuthnContextMapper` Attributes" on page 102

`SPAuthnContextMapper` **Parameters**

The following query parameters can be set in the URL when accessing `spSSOInit.jsp`:

- `AuthnContextClassRef` or `AuthnContextDeclRef`: These properties specify one or more URI references identifying the provider's supported authentication context classes. If a value is not specified, the default is `urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport`.
- *AuthLevel*: This parameter specifies the authentication level of the authentication context being used for authentication.
- *AuthComparison*: This parameter specifies the method of comparison used to evaluate the requested context classes or statements. Accepted values include:
  - *exact* where the authentication context statement in the assertion must be the exact match of, at least, one of the authentication contexts specified.

- *minimum* where the authentication context statement in the assertion must be, at least, as strong (as deemed by the identity provider) one of the authentication contexts specified.
- *maximum* where the authentication context statement in the assertion must be no stronger than any of the authentication contexts specified.
- *better* where the authentication context statement in the assertion must be stronger than any of the authentication contexts specified.

If the element is not specified, the default value is *exact*.

An example URL might be **`http://`***SP_host***`:`***SP_port***`/`***uri***`/spSSOInit.jsp?`**
`metaAlias=`*SP_MetaAlias***`&idpEntityID=`***IDP_EntityID***`&AuthnContextClassRef=PasswordProtectedTrans`**

`SPAuthnContextMapper` Attributes

The following attributes in the service provider extended metadata are used by the `SPAuthnContextMapper`:

- The `spAuthncontextMapper` property specifies the name of the service provider mapper implementation.
- The `spAuthncontextClassrefMapping` property specifies the map of authentication context class reference and authentication level in the following format:

  ```
  authnContextClassRef | authlevel [|| default]
  ```
- The `spAuthncontextComparisonType` property is optional and specifies the method of comparison used to evaluate the requested context classes or statements. Accepted values include:
  - *exact* where the authentication context statement in the assertion must be the exact match of, at least, one of the authentication contexts specified.
  - *minimum* where the authentication context statement in the assertion must be, at least, as strong (as deemed by the identity provider) one of the authentication contexts specified.
  - *maximum* where the authentication context statement in the assertion must be no stronger than any of the authentication contexts specified.
  - *better* where the authentication context statement in the assertion must be stronger than any of the authentication contexts specified.

  If the element is not specified, the default value is *exact*.

To Configure Mappings

The following procedure assumes you are mapping `urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport` to authentication level 4 on the service provider and use the LDAP authentication module for authentication on the identity provider.

1 **Set the mapping for the** `spAuthncontextClassrefMapping` **property in the current extended service provider metadata.**

For example, `PasswordProtectedTransport|4`

2 **Reload the modified metadata using** `saml2meta`**.**

See "The `saml2meta` Command-line Reference" on page 57.

3 **Set the mapping for the** `idpAuthncontextClassrefMapping` **property in the current extended identity provider metadata.**

For example,
`urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport|module=LDAP`

4 **Reload the modified metadata using** `saml2meta`**.**

See "The `saml2meta` Command-line Reference" on page 57.

5 **Access the single sign-on initialization page using the following URL:**

`http://`*AM_host***`:`***AM_port***`/`***uri***`/spSSOinit.jsp?`**
`metaAlias=/sp&idpEntityID=idp.sun.com&AuthnContextClassRef=PasswordProtectedTransport`

JavaServer Pages

JavaServer Pages (JSP) are HTML files that contain additional code to generate dynamic content. More specifically, they contain HTML code to display static text and graphics, as well as application code to generate information. When the page is displayed in a web browser, it will contain both the static HTML content and dynamic content retrieved via the application code. The SAML v2 Plug-in for Federation Services contains JSP that can initiate SAML v2 interactions. After installation, these pages can be accessed using the following URL format:

`http(s)://`*host***`:`***port***`/`***uri***`/saml2/jsp/`***jsp-page-name***`?metaAlias=`***xxx***`&`***...*

The JSP are collected in the `/`*AccessManager-base*`/`*product-directory*`/saml2/config/jsp` directory or the `/FederationManager-base/SUNWam/saml2/config/jsp` directory. The following sections contain descriptions of, and uses for, the JSP.

- "Default Display Page" on page 104
- "Assertion Consumer Page" on page 104
- "Single Sign-on Pages" on page 105
- "Name Identifier Pages" on page 107
- "Single Logout JavaServer Pages" on page 109

Caution – The following JSP cannot be modified:

- `idpArtifactResolution.jsp`
- `idpMNISOAP.jsp`
- `spMNISOAP.jsp`

Default Display Page

`default.jsp` is the default display page for the SAML v2 Plug-in for Federation Services. After a successful SAML v2 operation (single sign-on, single logout, or federation termination), a page is displayed. This page, generally the originally requested resource, is specified in the initiating request using the `<RelayState>` element. If a `<RelayState>` element is not specified, the value of the `<defaultRelayState>` property in the extended metadata configuration is displayed. If a `<defaultRelayState>` is not specified, this `default.jsp` is used. `default.jsp` can take in a message to display, for example, upon a successful authentication. The page can also be modified to add additional functionality.

Caution – When the value of `<RelayState>` or `<defaultRelayState>` contains special characters (such as `&`), it must be URL-encoded. For more information, see "Service Provider Extended Metadata Properties" on page 44.

Assertion Consumer Page

The `spAssertionConsumer.jsp` processes the responses that a service provider receives from an identity provider. When a service provider wants to authenticate a user, it sends an authentication request to an identity provider. The `AuthnRequest` asks that the identity provider return a `Response` containing one or more assertions. The `spAssertionConsumer.jsp` receives and parses the `Response` (or an artifact representing it). The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/Consumer`. Some ways in which the `spAssertionConsumer.jsp` can be customized include:

- The `localLoginUrl` parameter in the `spAssertionConsumer.jsp` retrieves the value of the `localAuthUrl` property in the service provider's extended metadata configuration. The value of `localAuthUrl` points to the local login page on the service provider side. If `localAuthUrl` is not defined, the login URL is calculated using the Assertion Consumer Service URL defined in the service provider's standard metadata configuration. Changing the `localLoginUrl` parameter value in `spAssertionConsumer.jsp` is another way to define the service provider's local login URL.
- After a successful single sign-on and before the final protected resource (defined in the `<RelayState>` element) is accessed, the user may be directed to an intermediate URL, if one is configured as the value of the `intermediateUrl` property in the service provider's

extended metadata configuration file. For example, this intermediate URL might be a successful account creation page after the auto-creation of a user account. The `redirectUrl` in `spAssertionConsumer.jsp` can be modified to override the `intermediateUrl` value.

Single Sign-on Pages

The single sign-on JSP are used to initiate single sign-on and, parse authentication requests, and generate responses. These include:

- "`idpSSOFederate.jsp`" on page 105
- "`idpSSOInit.jsp`" on page 105
- "`spSSOInit.jsp`" on page 106

`idpSSOFederate.jsp`

`idpSSOFederate.jsp` works on the identity provider side to receive and parse authentication requests from the service provider and generate a `Response` containing an assertion. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/idpSSOFederate`. `idpSSOFederate.jsp` takes the following parameters:

- `SAMLRequest`: This required parameter takes as a value the XML blob that contains the `AuthnRequest`.
- `metaAlias`: This optional parameter takes as a value the `metaAlias` set in the identity provider's extended metadata configuration file.
- `RelayState`: This optional parameter takes as a value the target URL of the request.

`idpSSOInit.jsp`

`idpSSOInit.jsp` initiates single sign-on from the identity provider side (also referred to as *unsolicited response*). For example, a user requests access to a resource. On receiving this request for access, `idpSSOInit.jsp` looks for a cached assertion which, if present, is sent to the service provider in an unsolicited `<Response>`. If no assertion is found, `idpSSOInit.jsp` verifies that the following required parameters are defined:

- `metaAlias`: This parameter takes as a value the `metaAlias` set in the identity provider's extended metadata configuration file. If the `metaAlias` attribute is not present, an error is returned.
- `spEntityID`: The entity identifier of the service provider to which the response is sent.

If defined, the unsolicited `Response` is created and sent to the service provider. If not, an error is returned. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/idpssoinit`. The following optional parameters can also be passed to `idpSSOInit.jsp`:

- `RelayState`: The target URL of the request.
- `NameIDFormat`: The currently supported name identifier formats: *persistent* or *transient*.

- `binding`: A URI suffix identifying the protocol binding to use when sending the `Response`. The supported values are:
  - `HTTP-Artifact`
  - `HTTP-POST`

`spSSOInit.jsp`

`spSSOInit.jsp` is used to initiate single sign-on from the service provider side. On receiving a request for access, `spSSOInit.jsp` verifies that the following required parameters are defined:

- `metaAlias`: This parameter takes as a value the `metaAlias` set in the identity provider's extended metadata configuration file. If the `metaAlias` attribute is not present, an error is returned.
- `idpEntityID`: The entity identifier of the identity provider to which the request is sent. If `idpEntityID` is not provided, the request is redirected to the SAML v2 IDP Discovery Service to get the user's preferred identity provider. In the event that more then one identity provider is returned, the last one in the list is chosen. If `idpEntityID` cannot be retrieved using either of these methods, an error is returned.

If defined, the `Request` is created and sent to the identity provider. If not, an error is returned. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/spssoinit`. The following optional parameters can also be passed to `spSSOInit.jsp`:

- `RelayState`: The target URL of the request.
- `NameIDFormat`: The currently supported name identifier formats: *persistent* or *transient*.
- `binding`: A URI suffix identifying the protocol binding to use when sending the `Response`. The supported values are:
  - `HTTP-Artifact`
  - `HTTP-POST`
- `AssertionConsumerServiceIndex`: An integer identifying the location to which the `Response` message should be returned to the requester. requester. It applies to profiles in which the requester is different from the presenter, such as the Web Browser SSO profile.
- `AttributeConsumingServiceIndex`: An integer indirectly specifying information (associated with the requester) describing the SAML attributes the requester desires or requires to be supplied.
- `isPassive`: Takes a value of `true` or `false` with `true` indicating the identity provider should authenticate passively.
- `ForceAuthN`: Takes a value of `true` indicating that the identity provider must force authentication or `false` indicating that the identity provider can reuse existing security contexts.
- `AllowCreate`: Takes a value of `true` indicating that the identity provider is allowed to created a new identifier for the principal if it does not exist or `false`.

- `Destination`: A URI indicating the address to which the request has been sent.
- `AuthnContextClassRef`: Specifies a URI reference identifying an authentication context class that describes the declaration that follows. Multiple references can be pipe-separated.
- `AuthnContextDeclRef`: Specifies a URI reference to an authentication context declaration. Multiple references can be pipe-separated.
- `AuthComparison`: The comparison method used to evaluate the requested context classes or statements. Accepted values include: *minimum*, *maximum* or *better*.
- `Consent`: Indicates whether or not (and under what conditions) consent has been obtained from a principal in the sending of this request.

Note – `Consent` is not supported in this release.

Name Identifier Pages

The various *ManageNameID* (MNI) JSP provide a way to change account identifiers or terminate mappings between identity provider accounts and service provider accounts. For example, after establishing a name identifier for use when referring to a principal, the identity provider may want to change its value and/or format. Additionally, an identity provider might want to indicate that a name identifier will no longer be used to refer to the principal. The identity provider will notify service providers of the change by sending them a `ManageNameIDRequest`. A service provider also uses this message type to register or change the `SPProvidedID` value (included when the underlying name identifier is used to communicate with it) or to terminate the use of a name identifier between itself and the identity provider.

- "`idpMNIRequestInit.jsp`" on page 107
- "`idpMNIRedirect.jsp`" on page 108
- "`spMNIRequestInit.jsp`" on page 108
- "`spMNIRedirect.jsp`" on page 109

`idpMNIRequestInit.jsp`

`idpMNIRequestInit.jsp` initiates the `ManageNameIDRequest` at the identity provider by user request. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/IDPMniInit`. It takes the following required parameters:

- `metaAlias`: The value of the `metaAlias` property set in the identity provider's extended metadata configuration file. If the `metaAlias` attribute is not present, an error is returned.
- `spEntityID`: The entity identifier of the service provider to which the response is sent.
- `requestType`: The type of `ManageNameIDRequest`. Accepted values include `Terminate` and `NewID`.

Note – `NewID` is not supported in this release.

Some of the other optional parameters are :

- `binding`: A URI specifying the protocol binding to use for the `<Request>`. The supported values are:
  - `urn:oasis:names:tc:SAML:2.0:bindings:SOAP`
  - `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
- `RelayState`: The target URL of the request

idpMNIRedirect.jsp

`idpMNIRedirect.jsp` processes the `ManageNameIDRequest` and the `ManageNameIDResponse` received from the service provider using HTTP-Redirect. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/IDPMniRedirect`. It takes the following required parameters:

- `SAMLRequest`: The `ManageNameIDRequest` from the service provider.
- `SAMLResponse`: The `ManageNameIDResponse` from the service provider.

Optionally, it can also take the `RelayState` parameter which specifies the target URL of the request.

spMNIRequestInit.jsp

`spMNIRequestInit.jsp` initiates the `ManageNameIDRequest` at the service provider by user request. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/SPMniInit`. It takes the following required parameters:

- `metaAlias`: This parameter takes as a value the `metaAlias` set in the identity provider's extended metadata configuration file. If the `metaAlias` attribute is not present, an error is returned.
- `idpEntityID`: The entity identifier of the identity provider to which the request is sent.
- `requestType`: The type of `ManageNameIDRequest`. Accepted values include `Terminate` and `NewID`.

Note – `NewID` is not supported in this release.

Some of the other optional parameters are :

- `binding`: A URI specifying the protocol binding to use for the `Request`. The supported values are:
  - `urn:oasis:names:tc:SAML:2.0:bindings:SOAP`

  - `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
- `RelayState`: The target URL of the request.

spMNIRedirect.jsp

`spMNIRedirect.jsp` processes the `ManageNameIDRequest` and the `<ManageNameIDResponse>` received from the identity provider using HTTP-Redirect. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/SPMniRedirect`. It takes the following required parameters:

- `SAMLRequest`: The `ManageNameIDRequest` from the identity provider.
- `SAMLResponse`: The `ManageNameIDResponse` from the identity provider.

Optionally, it can also take the `RelayState` parameter which specifies the target URL of the request.

Single Logout JavaServer Pages

The single logout JSP provides the means by which all sessions authenticated by a particular identity provider are near-simultaneously terminated. The single logout protocol is used either when a user logs out from a participant service provider or when the principal logs out directly from the identity provider.

- "`idpSingleLogoutInit.jsp`" on page 109
- "`idpSingleLogoutRedirect.jsp`" on page 110
- "`spSingleLogoutInit.jsp`" on page 110
- "`spSingleLogoutRedirect.jsp`" on page 111

idpSingleLogoutInit.jsp

`idpSingleLogoutInit.jsp` initiates a `LogoutRequest` at the identity provider by user request. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/IDPSloInit`. There are no required parameters. Optional parameters include:

- `RelayState`: The target URL after single logout.
- `binding`: A URI specifying the protocol binding to use for the `<Request>`. The supported values are:
  - `urn:oasis:names:tc:SAML:2.0:bindings:SOAP`
  - `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
- `Destination`: A URI indicating the address to which the request has been sent.
- `Consent`: Indicates whether or not (and under what conditions) consent has been obtained from a principal in the sending of this request.

Note – `Consent` is not supported in this release.

- `Extension`: Specifies permitted extensions as a list of string objects.

Note – `Extension` is not supported in this release.

- `logoutAll`: Specifies that the identity provider send log out requests to all service providers without a session index. It will logout all sessions belonging to the user.

`idpSingleLogoutRedirect.jsp`

`idpSingleLogoutRedirect.jsp` processes the `LogoutRequest` and the `LogoutResponse` received from the service provider using HTTP-Redirect. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/IDPSloRedirect`. It takes the following required parameters:

- `SAMLRequest`: The `LogoutRequest` from the service provider.
- `SAMLResponse`: The `LogoutResponse` from the service provider.

Optionally, it can also take the `RelayState` parameter which specifies the target URL of the request.

`spSingleLogoutInit.jsp`

`spSingleLogoutInit.jsp` initiates a `LogoutRequest` at the identity provider by user request. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/SPSloInit`. There are no required parameters. Optional parameters include:

- `RelayState`: The target URL after single logout.
- `binding`: A URI specifying the protocol binding to use for the `<Request>`. The supported values are:
  - `urn:oasis:names:tc:SAML:2.0:bindings:SOAP`
  - `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
- `Destination`: A URI indicating the address to which the request has been sent.
- `Consent`: Indicates whether or not (and under what conditions) consent has been obtained from a principal in the sending of this request.

Note – `Consent` is not supported in this release.

- `Extension`: Specifies permitted extensions as a list of string objects.

Note – `Extension` is not supported in this release.

`spSingleLogoutRedirect.jsp`

`spSingleLogoutRedirect.jsp` processes the `LogoutRequest` and the `LogoutResponse` received from the identity provider using HTTP-Redirect. The endpoint for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/SPSloRedirect`. It takes the following required parameters:

- `SAMLRequest`: The `LogoutRequest` from the identity provider.
- `SAMLResponse`: The `LogoutResponse` from the identity provider.

Optionally, it can also take the `RelayState` parameter which specifies the target URL of the request.

APPENDIX A

Deploying the SAML v2 Plug-in for Federation Services Generated WAR

The SAML v2 Plug-in for Federation Services runs in a simple web container and requires no complex integration with data stores or server environments. Once deployed, it extends the functionality of either server product to include SAML v2–based interactions. Instructions for deploying the SAML v2 Plug-in for Federation Services `WAR` in each of these web containers are in the following sections.

- "Deploying the SAML v2 Plug-in for Federation Services in Sun Java System Web Server" on page 113
- "Deploying the SAML v2 Plug-in for Federation Services in Sun Java System Application Server" on page 115
- "Deploying the SAML v2 Plug-in for Federation Services in BEA WebLogic Server" on page 118
- "Deploying Federation Manager in WebSphere Application Server" on page 120

Deploying the SAML v2 Plug-in for Federation Services in Sun Java System Web Server

Sun Java System Web Server has the `wdeploy` command line utility to deploy a WAR file. The syntax is:

```
wdeploy deploy -u /deployment-URI -i instance-name -v vs-id -d deployment-directory  WAR-file-location
```

where:

deployment-URI	Defines the URI for the application. **Note –** Note the inclusion of the leading slash.

instance-name	Defines the instance of Web Server to which you are deploying the web application.
vs-id	Defines the virtual server ID of the instance of Web Server to which you are deploying the web application.
deployment-directory	Defines the directory to which the WAR will be deployed.
WAR-file-location	Defines the full path to the WAR file being deployed.

When you execute `wdeploy`, a web application with the defined URI and directory is added to the `server.xml` file. The WAR file is also extracted from its location and deployed in the deployment directory.

Note – More information on the `wdeploy` utility can be found in the *Sun Java System Web Server 6.1 SP4 Programmer's Guide to Web Applications*.

To Deploy an Instance of the SAML v2 Plug-in for Federation Services in Web Server

To deploy the SAML v2 Plug-in for Federation Services in Web Server, type:

```
# WebServer-base/SUNWwbsvr/bin/https/httpadmin/bin/wdeploy deploy
-u /deployment-URI -i instance-name -v vs-id
-d WebServer-base/instance-name/deployment-URI  war-file-location
```

For example, when deploying the SAML v2 Plug-in for Federation Services in an instance of Federation Manager deployed in Web Server, you might use:

```
# /WebServer-base/SUNWwbsvr/bin/https/httpadmin/bin/wdeploy deploy -
u /saml2 -i niceday.red.sun.com -v https-niceday.red.sun.com
-d /opt/SUNWwbsvr/https-niceday.red.sun.com/saml2
/var/opt/SUNWam/fm/war_staging/federation.war
```

To Remove the SAML v2 Plug-in for Federation Services from Web Server

To remove the SAML v2 Plug-in for Federation Services from Web Server, type:

```
# /WebServer-base/bin/https/httpadmin/bin/wdeploy delete -u /deployment-URI
-i instance-name -v vs-id -n hard
```

where:

WebServer-base	Defines the Web Server installation directory.
deployment-URI	Defines the SAML v2 Plug-in for Federation Services URI (with leading slash).
instance-name	Defines the instance of Web Server to which the web application is deployed.
vs-id	Defines the virtual server ID of the instance of Web Server to which you are deploying the web application.

For example:

```
# /WebServer-base/SUNWwbsvr/bin/https/httpadmin/bin/wdeploy delete -u /saml2
-i https-niceday.red.sun.com -v https-niceday.red.sun.com -n hard
```

Deploying the SAML v2 Plug-in for Federation Services in Sun Java System Application Server

With Sun Java System Application Server, you can use the `deploy` subcommand of the `asadmin` utility to deploy a WAR file. The syntax is:

```
# asadmin deploy --user admin-user --passwordfile filename --port port
--contextroot deployment-URI --name deployment-URI
--target instance-name WAR-file-location
```

where:

admin-user	Defines the ID of the Application Server administrator.
filename	Defines the full path to the file that stores the password of the Application Server administrator. You must manually edit this file so it can be understood by the `asadmin` utility. The password must be in the form: `AS_ADMIN_PASSWORD=`*password* where *password* is the password in text used during the installation of Application Server.
port	Defines the port for the Application Server Domain Administration Server. The default is 4849.
deployment-URI	Defines the URI for the application. **Note –** Note the inclusion of the leading slash.
instance-name	Defines the instance of Application Server to which the WAR will be deployed.

WAR-file-location	Defines the full path to the WAR file being deployed.

Note – The `asadmin` options listed above are those relevant to deploying the SAML v2 Plug-in for Federation Services. For more information (including the full set of options), see the *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Reference Manual*.

To Deploy the SAML v2 Plug-in for Federation Services in Application Server

To deploy the SAML v2 Plug-in for Federation Services in Application Server, type:

```
# ApplicationServer-base/bin/asadmin deploy --user AS-administrator
--passwordfile filename --port port-number
--contextroot  deployment-URI --name deployment-URI
--target instance-name  war-file-location
```

For example, when deploying the SAML v2 Plug-in for Federation Services in an instance of Federation Manager deployed in Application Server, you might use:

```
# /opt/SUNWappserver/appserver/bin/asadmin deploy --user admin
--passwordfile /tmp/pwdfile --port 4849 --contextroot fm --name fm
--target server1 /var/opt/SUNWam/fm/war_staging/federation.war
```

Following the deployment, you must modify the Application Server `server.policy` file. By default, it is located in the `/var/opt/SUNWappserver/domains/`*domain-name*`/` directory. In the sample below, the capitalized contents (all but `WEB-INF`) must be replaced with information applicable to your deployment.

EXAMPLE A–1 Application Server `server.policy` File

```
// Federation Manager RELATED ADDITIONS
   grant {
     permission java.util.PropertyPermission "user.language", "write";
   };
   grant codeBase "file:${BASEDIR}/${PROD_DIR}/fm/web-src/WEB-INF/lib/am_sdk.jar" {
     permission java.net.SocketPermission "*", "connect,accept,resolve";
   };
   grant codeBase "file:${BASEDIR}/${PROD_DIR}/fm/web-src/WEB-INF/lib/am_services.jar" {
        permission java.net.SocketPermission "*", "connect,accept,resolve";
   };
   grant codeBase "file:$AS81_VARDIR/domains/$AS81_DOMAIN/applications/
   j2ee-modules/${DEPLOY_WARPREFIX}/-" {
        permission java.net.SocketPermission "*", "connect,accept,resolve";
```

EXAMPLE A–1 Application Server `server.policy` File *(Continued)*

```
};
grant {
 permission java.lang.RuntimePermission "modifyThreadGroup";
 permission java.lang.RuntimePermission "setFactory";
 permission java.lang.RuntimePermission "accessClassInPackage.*";
 permission java.util.logging.LoggingPermission "control";
 permission java.lang.RuntimePermission "shutdownHooks";
 permission javax.security.auth.AuthPermission "insertProvider.Mozilla-JSS";
 permission java.security.SecurityPermission "putProviderProperty.Mozilla-JSS";
 permission javax.security.auth.AuthPermission "getLoginConfiguration";
 permission javax.security.auth.AuthPermission "setLoginConfiguration";
 permission javax.security.auth.AuthPermission "modifyPrincipals";
 permission javax.security.auth.AuthPermission "createLoginContext.*";
 permission java.security.SecurityPermission "insertProvider.Mozilla-JSS";
 permission javax.security.auth.AuthPermission "putProviderProperty.Mozilla-JSS";
 permission java.io.FilePermission "ALL FILES", "execute,delete";
 permission java.io.FilePermission "$VAR_SUBDIR/logs/*", "delete,write";
 permission java.util.PropertyPermission "java.util.logging.config.class", "write";
 permission java.security.SecurityPermission "removeProvider.SUN";
 permission java.security.SecurityPermission "insertProvider.SUN";
 permission java.security.SecurityPermission "removeProvider.Mozilla-JSS";
 permission javax.security.auth.AuthPermission "doAs";
 permission java.util.PropertyPermission "java.security.krb5.realm", "write";
 permission java.util.PropertyPermission "java.security.krb5.kdc", "write";
 permission java.util.PropertyPermission "java.security.auth.login.config", "write";
 permission javax.security.auth.kerberos.ServicePermission "*", "accept";
 permission javax.net.ssl.SSLPermission "setHostnameVerifier";
};
```

Modifications to `server.policy` are made as follows:

TABLE A–1 `server.policy` Modifications After Installation

Replaceable Content	Default Value
`$BASEDIR`	`/opt`
`$PROD_DIR`	`SUNWam`
`$AS81_VARDIR`	`/var/opt/SUNWappserver`
`$AS81_DOMAIN`	`domain1`
`$VAR_SUBDIR`	`/var/opt/SUNWam`
`$DEPLOY_WARPREFIX`	`federation`

To Remove the SAML v2 Plug-in for Federation Services from Application Server

To remove the SAML v2 Plug-in for Federation Services from Application Server, type:

```
# ApplicationServer-base/bin/asadmin undeploy --user AS-administrator
--passwordfile filename --host host --port port
 --target instance-name  deployment-URI
```

where:

ApplicationServer-base	Defines the Application Server installation directory.
AS-administrator	Defines the administrator of the Application Server
filename	Defines the file that holds the Application Server administrator password.
host	Defines the host server being used.
port	Defines the port number being used.
instance-name	Defines the instance of Application Server to which the application is deployed.
deployment-URI	Defines the SAML v2 Plug-in for Federation Services URI (with leading slash).

For example:

```
# /opt/SUNWappserver/appserver/bin/asadmin undeploy --user admin
--passwordfile /tmp/pwdfile --host samplehost.sun.com
--port 4849 --target server1 /saml2
```

Deploying the SAML v2 Plug-in for Federation Services in BEA WebLogic Server

With BEA WebLogic Server 8.1 (minimum version supported), you can use the `weblogic.Deployer` utility on the command line to deploy a WAR file. More information on this utility can be found in the BEA WebLogic Server documentation.

To Deploy the SAML v2 Plug-in for Federation Services in BEA WebLogic Server

To deploy the SAML v2 Plug-in for Federation Services in WebLogic Server, type:

```
# WebLogic-base/bin/java -classpath WebLogic-classpath weblogic.Deployer
-adminurl http://server-host:server-port -name deployment-URI
-username WebLogic-admin -password WebLogic-admin-password -target instance-name
-deploy WAR-file-location
```

where:

WebLogic-base	Defines the WebLogic Server installation directory.
WebLogic-classpath	Includes the JDK path and weblogic.jar.
server-host	Defines the WebLogic Server host machine.
server-port	Defines the port of the WebLogic Server host machine.
deployment-URI	Defines the SAML v2 Plug-in for Federation Services URI.
WebLogic-admin	Defines the ID of the WebLogic Server super user (weblogic, by default).
WebLogic-admin-password	Defines the password of the WebLogic Server super user.
instance-name	Defines the instance of WebLogic Server to which the WAR will be deployed.
WAR-file-location	Defines the full path to the WAR file being deployed.

For example, when deploying the SAML v2 Plug-in for Federation Services in an instance of Federation Manager deployed in the WebLogic Server, you might use:

```
/export/bea8/jdk142_06/bin/java -classpath /export/bea8/weblogic8/lib/weblogic.jar:.
weblogic.Deployer -adminurl http://samplehost.sun.com:7001 -name /saml2
-username weblogic -password 11111111 -target myserver
-deploy /var/opt/SUNWam/fm/war_staging/federation.war
```

To Remove the SAML v2 Plug-in for Federation Services from BEA WebLogic Server

To remove the SAML v2 Plug-in for Federation Services from WebLogic Server, type:

```
# WebLogic-base/bin/java -classpath WebLogic-classpath weblogic.Deployer -undeploy
-adminurl http://server-host:server-port -name deployment-URI
-username WebLogic-admin -password WebLogic-admin-password
-target WebLogic-Server
```

where:

WebLogic-base	Defines the WebLogic Server installation directory.
WebLogic-classpath	Includes the JDK path and weblogic.jar.
server-host	Defines the WebLogic Server host machine.
server-port	Defines the port of the WebLogic Server host machine.
deployment-URI	Defines the SAML v2 Plug-in for Federation Services URI.
WebLogic-admin	Defines the ID of the WebLogic Server super user (weblogic, by default).
WebLogic-admin-password	Defines the password of the WebLogic Server super user.
WebLogic-Server	Defines the instance of WebLogic Server.

For example:

```
/export/bea8/jdk142_06/bin/java -classpath /export/bea8/weblogic8/lib/weblogic.jar:.
weblogic.Deployer -undeploy -adminurl http://samplehost.sun.com:7001
-name /saml2 -username weblogic -password 11111111 -target myserver
```

Deploying Federation Manager in WebSphere Application Server

Before deploying an application or deleting an instance in WebSphere Application Server 5.1 (minimum version supported), you must modify the `Jacl` (Java Action Command Language) descriptor file. Following this you may run the wsadmin.sh file from the command line. More information on both of these steps can be found in the WebSphere Application Server documentation.

To Deploy the SAML v2 Plug-in for Federation Services in WebSphere Application Server

To deploy the SAML v2 Plug-in for Federation Services in WebSphere Application Server, edit the Jacl descriptor by adding the following:

```
\$AdminApp install WAR-file-location {-contextroot deployment-URI
-usedefaultbindings -nopreCompile JSPs -distributeApp
-nouseMetaDataFromBinary -node WebSphereAS-node -cell WebSphereAS-cell -server
WebSphereAS-instance -nodeployejb -appname deployment-URI
-createMBeansForResources -noreloadEnabled
-reloadInterval 0 -nodeployws}
```

where:

WAR-file-location	Defines the full path to the WAR file being deployed.
deployment-URI	Defines the SAML v2 Plug-in for Federation Services URI.
WebSphereAS-node	Defines the node under which the Application Server instance is configured.
WebSphereAS-cell	Defines the cell under which the Application Server node is configured.
WebSphereAS-instance	Defines the instance of the Application Server to which the SAML v2 Plug-in for Federation Services will be deployed.

After editing the Jacl descriptor, run the following command to deploy the WAR:

```
# WebSphereAS-base/bin/wsadmin.sh -f Jacl-descriptor-file
```

To Remove the SAML v2 Plug-in for Federation Services from WebSphere Application Server

To remove the SAML v2 Plug-in for Federation Services from WebSphere Application Server, edit the `Jacl` descriptor by adding the following:

```
\$AdminApp uninstall deploy-tag {-node WebSphereAS-node -cell WebSphereAS-cell
-server WebSphereAS-instance}
```

where:

deploy-tag	Defines the SAML v2 Plug-in for Federation Services URI.
WebSphereAS-node	Defines the node under which the Application Server instance is configured.
WebSphereAS-cell	Defines the cell under which the Application Server node is configured.
WebSphereAS-instance	Defines the instance of the Application Server to which Federation Manager will be deployed.

After editing the `Jacl` descriptor, run the following command to remove:

```
# WebSphereAS-base/bin/wsadmin.sh -f Jacl-descriptor-file
```

APPENDIX B

Log Message Reference

This appendix lists the possible log messages for each functional area of the SAML v2 Plug-in for Federation Services. The default location for SAML v2 logs in Access Manager is `/var/opt/`*product-directory*`/logs`. The default location for SAML v2 logs in Federation Manager is `/var/opt/SUNWam/fm/logs`.

Note – Metadata is sometimes referred to as *entity descriptor* or *entity configuration* where entity generically refers to the `entityID` with which each provider is given a unique identifier. For more information on the `entityID`, see "Extended Metadata Properties" on page 41.

The table in this appendix documents the following log file items:

Id	The log identification number prefixed in the log itself by *SAML2–*
Description	The log message
Data	The data type to which the message pertains
Trigger	Reason for the message
Action	Possible corrective action

Note – Please note the following:

- *Setting entity configuration* refers to using `saml2meta` command to load an extended metadata configuration file.
- *Creating entity descriptor* refers to using `saml2meta` command to load either a standard or an extended metadata configuration file.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services

Id	Description	Data	Trigger	Action
1	Invalid service provider entity identifier	Service provider entity identifier	Cannot process request.	Check the service provider entity identifier.
2	Invalid identity provider entity identifier	Identity provider entity identifier	Cannot process request.	Check the identity provider entity identifier.
3	Unable to retrieve service provider metadata.	Service provider entity identifier	Cannot retrieve service provider metadata.	■ Check that the data store is accessible. ■ Check the realm or organization name. ■ Check the service provider entity identifier.
4	Unable to retrieve identity provider metadata.	Identity provider entity identifier	Cannot retrieve identity provider metadata.	■ Check that the data store is accessible. ■ Check the realm or organization name. ■ Check the identity provider entity identifier.
5	Unable to retrieve Single Sign-on Service URL.	Identity provider entity identifier	Error retrieving Single Sign-on Service URL.	■ Check that the data store is accessible. ■ Check the realm or organization name. ■ Check the identity provider entity identifier.
6	Redirecting to Single Sign On Service	Single Sign On Service URL	Sending Authentication Request by redirecting to Single Sign-on Service URL.	
7	Unable to retrieve a response using Response ID after local login.	Response ID	Response doesn't exist in the service provider cache.	Check the service provider's cache clean up interval configuration.
8	Unable to retrieve an artifact from HTTP Request.	Artifact	`<SAMLart>` is missing from HTTP Request	■ Check with sender. ■ Check web container server log.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
9	Received artifact from HTTP Request.	Artifact value	Received `<SAMLart>` from HTTP Request in the process of single sign-on using Artifact Profile.	
10	Unable to find identity provider entity identifier based on the value of the `<SourceID>` element in artifact.	■ Artifact value ■ Realm or organization name	No matching identity provider entity identifier found in metadata configuration.	Check that identity provider's metadata is loaded.
11	Unable to load identity provider's metadata.	■ Identity provider entity identifier ■ Realm or organization name	Unable to load identity provider's metadata.	■ Check that the identity provider's metadata is configured correctly. ■ Check the realm or organization name. ■ Check the identity provider entity identifier.
12	Unable to find the identity provider's Artifact Resolution Service URL.	Identity provider entity identifier	Artifact Resolution Service URL is not defined in identity provider's metadata.	Check that the identity provider's Artifact Resolution Service URL is defined in the standard metadata.
13	Unable to create `<ArtifactResolve>` element.	■ Hosted service provider entity identifier ■ Value of artifact	Error when creating `<Artifact Resolve>` instance.	Check implementation of Artifact Resolution Service URL.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
14	Unable to obtain SOAP response from identity provider's Artifact Resolution Service URL.	■ Hosted service provider entity identifier ■ Identity provider's Artifact Resolution Service URL	Error in SOAP communication.	■ Check identity provider's Artifact Resolution Service URL. ■ Check SOAP message's authentication requirements against those for identity provider's Artifact Resolution Service.
15	Obtained response using artifact profile.	■ Hosted service provider entity identifier ■ Remote identity provider entity identifier ■ Artifact value ■ XML response string (if log level was set to `LL_FINE` at run time)	Single Sign On using Artifact Profile.	
16	Unable to obtain Artifact Response due to SOAP error.	Identity provider entity identifier	Error in SOAP communication.	Check identity provider configuration.
17	Received SOAP Fault instead of `<ArtifactResponse>`.	Identity provider entity identifier	Error in identity provider's Artifact Resolution Service.	■ Check identity provider's Artifact Resolution Service URL. ■ Check debug file for detailed information.
18	Received too many artifact responses.	Identity provider entity identifier	Identity provider sent more than one `<Artifact Response>` in SOAP message.	Check identity provider configuration.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
19	Unable to instantiate `<ArtifactResponse>`.	Identity provider entity identifier	Error while instantiating `<Artifact Response>`.	■ Check identity provider configuration. ■ Check debug file for detailed information.
20	Unable to obtain `<ArtifactResponse>` from SOAP message.	Identity provider entity identifier	No `<Artifact Response>` is included in SOAP message.	Check identity provider configuration.
21	Unable to verify `<ArtifactResponse>` signature.	Identity provider entity identifier	Error while trying to verify signature on `<Artifact Response>`.	■ Check identity provider configuration. ■ Check debug file for detailed information.
22	Invalid `InResponseTo` attribute in `<ArtifactResponse>`.	Identity provider entity identifier	`InResponseTo` attribute in `<Artifact Response>` is missing or doesn't match Artifact Resolve ID.	Check identity provider configuration.
23	Invalid issuer in `<ArtifactResponse>`.	Identity provider entity identifier	Issuer in `<Artifact Response>` is missing or doesn't match with identity provider entity identifier.	Check identity provider configuration.
24	Invalid status code in `<ArtifactResponse>`.	■ Identity provider entity identifier ■ Status code (if log level was set to `LL_FINE` at run time)	Status in `<Artifact Response>` is missing or status code is not `Success`.	Check identity provider configuration.
25	Unable to instantiate responses from `<ArtifactResponse>`.	Identity provider entity identifier	Error occurred while instantiating `<Response>`.	Check debug file for detailed information.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
26	`<SAMLResponse>` is missing from HTTP post interaction.		`<SAMLResponse>` attribute is missing from HTTP POST.	
27	Unable to instantiate response from POST.		Error occurred while instantiating `<Response>`.	Check debug file for detailed information.
28	Unable to decode `<Response>`.		Error occurred while decoding `<Response>`.	Check debug file for detailed information.
29	Obtained response using POST profile.	XML string (if the log level was set to `LL_FINE` at run time)	Successful single sign-on using POST Profile.	
30	Wrote federation information.	■ User name ■ `<NameIDInfo>` (if log level was set to `LL_FINE` at run time)	Successful user federation.	
31	Redirect request to identity provider.	Redirection URL	Single logout.	
32	Unable to find Assertion Consumer Service URL.	`metaAlias`	Single sign-on.	
33	Unable to find return binding.	`metaAlias`	Single sign-on.	
34	Unable to post a response to target.	Assertion Consumer Service URL	Single sign-on with POST binding.	
35	Unable to create an artifact.	Identity provider entity identifier	Single sign-on with Artifact binding.	

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
36	Received `<AuthnRequest>`.	■ Service provider entity identifier ■ Identity provider `metaAlias` ■ `<AuthnRequest>` XML string	Single sign-on.	
37	Post response to service provider.	■ Service provider entity identifier ■ Identity provider `metaAlias` ■ `<Response>` XML string	Single sign-on with POST binding.	
38	Send an artifact to SP.	■ Identity provider entity identifier ■ Identity provider realm or organization ■ Redirect URL	Single sign-on with Artifact binding.	
39	Encountered invalid SOAP message error on identity provider.	Identity provider entity identifier	Single sign-on with Artifact binding.	
40	Artifact response sent to service provider.	■ Identity provider entity identifier ■ Artifact string ■ Artifact response	Single sign-on with Artifact binding.	

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
41	Entity descriptor obtained.	■ Entity identifier ■ Realm or organization	Obtain entity descriptor.	
42	Encountered invalid realm or organization error while getting entity descriptor.	Realm or organization name	Obtain entity descriptor.	Check the realm or organization name.
43	Obtained invalid entity descriptor.	■ Entity identifier ■ Realm or organization name	Obtain entity descriptor.	Delete invalid metadata and import it again.
44	Encountered configuration error while getting entity descriptor.	■ Error message ■ Entity identifier ■ Realm or organization name	Obtain entity descriptor.	Check debug file for detailed information.
45	No entity identifier found.	Realm or organization name	Set entity descriptor.	Set entity identifier in provider metadata.
46	Invalid realm or organization error while setting entity descriptor.	Realm or organization name	Set entity descriptor.	Check the realm or organization name.
47	Entity descriptor doesn't exist.	■ Entity identifier ■ Realm or organization name	Set entity descriptor.	Create metadata for provider.
48	Entity descriptor was set.	■ Entity identifier ■ Realm or organization name	Set entity descriptor.	

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
49	Encountered configuration error while setting entity descriptor.	■ Error message ■ Entity identifier ■ Realm or organization name	Set entity descriptor.	Check debug file for detailed information.
50	Invalid entity descriptor to set.	■ Entity identifier ■ Realm or organization name	Set entity descriptor.	Check entity descriptor if it follows the schema.
51	No entity identifier found while creating entity descriptor.	Realm or organization name	Creating entity descriptor.	Set entity identifier in entity descriptor.
52	Invalid realm found while creating entity descriptor.	Realm or organization name	Creating entity descriptor.	Check the realm name.
53	Configured entity descriptor found already found.	■ Entity identifier ■ Realm or organization name	Creating entity descriptor.	Delete existing entity descriptor before configuring new one.
54	Entity descriptor successfully created.	■ Entity identifier ■ Realm or organization name	Creating entity descriptor.	
55	Configuration error occurred when creating entity descriptor.	■ Error message ■ Entity identifier ■ Realm or organization name	Creating entity descriptor.	Check debug file for detailed information.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
56	Cannot create entity descriptor.	■ Entity identifier ■ Realm or organization name	Creating entity descriptor.	Check that the XML syntax of the entity descriptor follows the schema.
57	Invalid realm error occurred when deleting entity descriptor.	Realm or organization name	Deleting entity descriptor.	Check the realm or organization name.
58	Entity descriptor doesn't exist.	■ Entity identifier ■ Realm or organization name	Deleting entity descriptor.	
59	Entity descriptor was successfully deleted.	■ Entity identifier ■ Realm or organization name	Deleting entity descriptor.	
60	Configuration error while deleting entity descriptor.	■ Error message ■ Entity identifier ■ Realm or organization name	Deleting entity descriptor.	Check debug file for detailed information.
61	Successfully retrieved entity configuration.	■ Entity identifier ■ Realm or organization name	Getting entity configuration.	
62	Invalid realm or organization error while getting entity configuration.	Realm or organization name	Getting entity configuration.	Check the realm or organization name.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
63	Retrieved invalid entity configuration.	■ Entity identifier ■ Realm or organization name	Getting entity configuration.	Delete invalid entity configuration and import it again.
64	Configuration error while getting entity configuration.	■ Error message ■ Entity identifier ■ Realm or organization name	Getting entity configuration.	Check debug file for detailed information.
65	No entity identifier value found while setting entity configuration.	Realm or organization name	Setting entity configuration.	Define entity identifier in entity configuration.
66	Invalid realm value found while setting entity configuration.	Realm or organization name	Setting entity configuration.	Check the realm or organization name.
67	Entity configuration doesn't exist.	■ Entity identifier ■ Realm or organization name	Setting entity configuration.	Create an entity descriptor before setting entity configuration.
68	Entity configuration was successfully set.	■ Entity identifier ■ Realm or organization name	Setting entity configuration.	
69	Configuration error occurred while setting entity configuration.	■ Error message ■ Entity identifier ■ Realm or organization name	Setting entity configuration.	Check debug file for detailed information.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
70	Referenced entity configuration is invalid.	■ Entity identifier ■ Realm or organization name	Setting entity configuration.	Check that the XML syntax of the entity descriptor follows the schema.
71	No entity identifier found while creating entity configuration.	Realm or organization name	Creating entity configuration.	Define entity identifier in entity configuration.
72	Invalid realm or organization value while creating entity configuration.	Realm or organization name	Creating entity configuration.	Check the realm or organization name.
73	Entity descriptor doesn't exist while creating entity configuration.	■ Entity identifier ■ Realm or organization name	Creating entity configuration.	Create entity descriptor before creating entity configuration.
74	Entity configuration already exists.	■ Entity identifier ■ Realm or organization name	Creating entity configuration.	Delete existing entity configuration first.
75	Entity configuration was successfully created.	■ Entity identifier ■ Realm or organization name	Creating entity configuration.	
76	Configuration error occurred while creating entity configuration.	■ Error message ■ Entity identifier ■ Realm or organization name	Creating entity configuration.	Check debug file for detailed information.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
77	Entity configuration cannot be created.	■ Entity identifier ■ Realm or organization name	Creating entity configuration.	Check that the XML syntax of the entity descriptor follows the schema.
78	Invalid realm or organization value while deleting entity configuration.	Realm or organization name	Deleting entity configuration.	Check the realm or organization name.
79	Entity configuration doesn't exist.	■ Entity identifier ■ Realm or organization name	Deleting entity configuration.	Check debug file for detailed information.
80	Entity configuration was successfully deleted.	■ Entity identifier ■ Realm or organization name	Deleting entity configuration.	
81	Configuration error occurred while deleting entity configuration.	■ Error message ■ Entity identifier ■ Realm or organization name	Deleting entity configuration.	Check debug file for detailed information.
82	Value of realm or organization is invalid.	Realm or organization name	Retrieving all hosted entities.	Check the realm or organization name.
83	Configuration error occurred while retrieving all hosted entities.	■ Error message ■ Realm or organization name	Retrieving all hosted entities.	Check debug file for detailed information.
84	Successfully retrieved all hosted entities.	■ Error message ■ Realm or organization name	Retrieving all hosted entities.	
85	Value of realm or organization is invalid.	Realm or organization name	Retrieving all remote entities.	Check the realm or organization name.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
86	Configuration error occurred while retrieving all remote entities.	■ Error message ■ Realm or organization name	Retrieving all remote entities.	Check debug file for detailed information.
87	Successfully retrieved all remote entities.	■ Error message ■ Realm or organization name	Retrieving all remote entities.	
88	`InResponseTo` attribute in response is invalid.	Response ID	Service provider receiving a response to single sign-on request.	Check debug file for detailed information.
89	Issuer in response is invalid, not configured or not trusted by the hosted provider.	■ Hosted entity identifier ■ Realm or organization name ■ Response identifier	Service provider receiving a response to single sign-on request.	Check configuration.
90	Status code in response does not denote success.	`<ResponseID>` status code (if log level was set to `LL_FINE` at run time)	Service provider receiving a response to single sign-on request.	Most likely an error occurred at identity provider. Check the status code and contact identity provider, if needed.
91	Assertion in response was not encrypted.	Response ID	Service provider requested that the response's assertion be encrypted, but it received an assertion(s) that was (were) not encrypted.	■ Check configuration. ■ Notify identity provider regarding the requirement.
92	Response has no assertion.	Response ID	Service provider received a single sign-on response that contained no assertion.	Check error code of the response and notify identity provider, if needed.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
93	Issuer in assertion is invalid, not configured at service provider, or not trusted by the service provider.	Assertion ID	Service provider receiving a response to single sign-on request.	Check configuration
94	Issuer in assertion did not match the issuer in the response or other assertions in the response.	Assertion ID	Service provider receiving a response to single sign-on request.	Check debug file for detailed information.
95	Assertion was not signed or the signature is not valid.	Assertion ID	Service provider requested that the assertion in a response to a single sign-on request be signed but it received an assertion(s) that was (were) not signed or one that contained an invalid signature.	■ Check configuration. ■ Check debug file for detailed information.
96	`<SubjectConfirmationData>` element had no subject.	Assertion ID	Service provider received an assertion in a response to a single sign-on request.	■ Check debug file for the assertion received. ■ Contact identity provider, if needed.
97	`<SubjectConfirmationData>` element had no recipient.	Assertion ID	Service provider received an assertion in a response to a single sign-on request.	■ Check debug file for the assertion received. ■ Contact identity provider, if needed.
98	Service provider that received the response is not the intended recipient.	Assertion ID	Service provider received an assertion in a response to a single sign-on request.	■ Check debug file for the assertion received. ■ Check metadata. ■ Contact identity provider, if needed.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
99	Time defined in `<SubjectConfirmationData>` element is invalid.	Assertion ID	Service provider received an assertion in a response to a single sign-on request.	■ Synchronize the time between service provider and identity provider. ■ Increase the time skew attribute in the service provider entity configuration.
100	Assertion received by service provider had a time defined in the `<notBefore>` attribute of `<Subject ConfirmationData>` element.	Assertion ID	Service provider received an assertion in a response to a single sign-on request.	■ Check debug file for the assertion received. ■ Contact identity provider, if needed.
101	Assertion received by service provider contained a different `<InResponseTo>` attribute from the one in the response, or it contained no `<InResponseTo>` value but the response did.	Assertion ID	Service provider received an assertion in a response to a single sign-on request.	■ Check debug file for the assertion received. ■ Contact identity provider, if needed.
102	Assertion received by service provider contained no conditions.	Assertion ID	Service provider received an assertion in a response to a single sign-on request.	■ Check debug file for the assertion received. ■ Contact identity provider, if needed.
103	Assertion received by service provider contained no `<Audience Restriction>`.	Assertion ID	Service provider received an assertion in a response to a single sign-on request.	■ Check debug file for the assertion received. ■ Contact identity provider, if needed.
104	Assertion received by service provider was not the intended audience of the single sign-on assertion.	Assertion ID	Service provider received an assertion in a response to a single sign-on request.	■ Check debug file for the assertion received. ■ Check metadata. ■ Contact identity provider, if needed.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
105	Successfully found authentication assertion in the response.	■ Assertion ID ■ `<Subject>` (if log level was set to `LL_FINE` at run time) ■ SessionIndex, if any	Both the response and assertion(s) inside the response are valid.	
106	Invalid single sign-on token was found in the request.	Single sign-on token value	Provider initiated single logout without a single sign-on token.	
107	No entity identifier is specified in the request.	Entity identifier value	Provider initiated a request without an entity identifier.	Specify `EntityID` parameter in request URL.
108	No `metaAlias` is specified in a request.	Value of `metaAlias`	Provider initiated a request without `metaAlias`.	Specify `metaAlias` parameter in request URL.
109	Successful redirection of request to authentication page.	URL to Authentication page	Request initiated without a single sign-on token.	
110	Provider cannot decode URL encoded Query parameter.	URL encoded Query parameter	Initiate to decode incorrectly URL encoded Query parameter.	
111	Cannot instantiate MNI Response because of incorrect XML value.	XML value of MNI Response	Provider can't parse MNI Response with incorrect XML string.	
112	Cannot instantiate MNI Request because of incorrect XML value.	XML value of MNI Request	Provider can't parse MNI Request with incorrect XML string.	
113	Cannot instantiate single logout Response because of incorrect XML value.	XML value of single logout Response	Provider can't parse single logout Response with incorrect XML string.	

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
114	Cannot instantiate single logout request because of incorrect XML value.	XML value of single logout request	Provider can't parse single logout request with incorrect XML string.	
115	Cannot verify signature in MNI request.	MNI request with signature	Signature in MNI request is incorrect.	
116	Cannot verify signature in MNI response.	MNI response with signature	Signature in MNI response is incorrect.	
117	Cannot verify signature in single logout request.	Single logout request with signature	Signature in single logout request is incorrect.	
118	Cannot verify signature in single logout response.	Single logout response with signature	Signature in single logout response is incorrect.	
119	Cannot decrypt `<EncryptedID>`.	Exception message	Provider attempts to decrypt an encrypted `<EncryptedID>`.	
120	MNI response throws error.	Status message	MNI request caused problem.	
121	Single logout response throws error.	Status message	Single logout request caused problem.	
122	Entity role is not specified in the request.	Entity Role value	Initiated request contains no Entity Role value.	Specify Entity Role parameter in the request.
123	Successfully retrieved circle of trust from cache.	■ Name of the circle of trust ■ Realm or organization name	Obtained the circle of trust from cache.	

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
124	Invalid realm while getting the circle of trust.	Realm or organization name	Obtain the circle of trust	Check the realm or organization value.
125	Obtained the circle of trust from directory.	■ Name of the circle of trust ■ Realm or organization name	Obtain the circle of trust	
126	Error while retrieving circle of trust from directory.	■ Error message ■ Name of the circle of trust ■ Realm or organization name	Obtain the circle of trust	■ Check configuration. ■ Check debug file for detailed information.
127	Invalid realm value found while retrieving all circles of trust.	Realm or organization name	Getting all circles of trust.	Check the realm or organization value.
128	Error while retrieving all circles of trust.	■ Error message ■ Search pattern ■ Realm or organization name	Getting all circles of trust.	■ Check configuration. ■ Check debug file for detailed information.
129	Invalid name identifier throws error while modifying circle of trust.	Realm or organization name	Modifying the circle of trust.	Check the name identifier of circle of trust descriptor.
130	Invalid realm or organization name while modifying circle of trust.	Realm or organization name	Modifying the circle of trust.	Check the realm or organization value.
131	Modified circle of trust.	■ Name of the circle of trust ■ Realm or organization name	Modifying the circle of trust.	

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
132	Error while modifying the circle of trust.	■ Error message ■ Name of the circle of trust ■ Realm or organization name	Modifying the circle of trust.	Check debug for more detailed error message.
133	Invalid name identifier throws error while creating circle of trust.	Realm or organization name	Creating circle of trust.	Check the name identifier of circle of trust descriptor.
134	Circle of trust being created already exists.	■ Name of the circle of trust ■ Realm or organization name	Creating circle of trust.	Check the name identifier of circle of trust descriptor.
135	Invalid realm or organization error while creating the circle of trust.	Realm or organization name	Creating circle of trust.	Check the realm or organization value.
136	Circle of trust successfully created.	■ Name of the circle of trust ■ Realm or organization name	Creating circle of trust.	
137	Invalid realm or organization error while deleting the circle of trust.	Realm or organization name	Deleting circle of trust.	Check the realm value.
138	Circle of trust successfully deleted.	■ Name of the circle of trust ■ Realm or organization name	Deleting circle of trust.	
139	Error while deleting the circle of trust.	■ Error message ■ Name of the circle of trust ■ Realm or organization name	Deleting circle of trust.	Check debug file for more detailed information.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
140	Invalid realm or organization error while getting list of all active circles of trust.	Realm or organization name	Retrieving list of all active circles of trust.	Check the realm or organization value.
141	Error thrown while getting list of all active circles of trust.	■ Error message ■ Realm or organization name	Getting all active circle of trust.	Check debug file for more detailed information.
142	Invalid name identifier error thrown.	Realm or organization name	Adding member to circle of trust.	Check the name of the circle of trust.
143	No entity identifier found while adding member to circle of trust.	Realm or organization name	Adding member to the circle of trust.	Check the entity identifier.
144	Invalid realm or organization error thrown.	Realm or organization name	Adding member to the circle of trust.	Check the realm or organization value.
145	Configuration error thrown.	■ Error message ■ Name of the circle of trust ■ Entity identifier ■ Realm or organization name	Adding member to the circle of trust.	Check debug file for more detailed information.
146	Invalid name identifier error thrown.	Realm or organization name	Removing member from circle of trust.	Check the name of the circle of trust.
147	No entity identifier found.	■ Name of the circle of trust ■ Realm or organization name	Removing member from circle of trust.	Check the entity identifier in metadata.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
148	Configuration error thrown.	■ Error message ■ Name of the circle of trust ■ Entity identifier ■ Realm or organization name	Removing member from circle of trust.	Check debug file for more detailed information.
149	Invalid realm or organization found.	Realm or organization name	Listing trusted providers in circle of trust.	Check the realm or organization value.
150	Configuration error thrown.	■ Error message ■ Name of the circle of trust ■ Realm or organization name	Listing trusted providers in circle of trust.	Check debug file for more detailed information.
151	Invalid realm or organization error thrown.	Realm or organization name	Removing trusted provider from circle of trust.	Check the realm or organization value.
152	Configuration error thrown.	■ Error message ■ Name of the circle of trust ■ Entity identifier ■ Realm or organization name	Determining if entity is member of circle of trust.	Check debug file for more detailed information.
153	Issuer in request is invalid.	■ Hosted entity identifier ■ Realm or organization name ■ Request ID	Issuer in request is not configured or not trusted by the hosted provider	Check configuration.
154	Invalid realm or organization error thrown.	Realm or organization name	Retrieving all entities.	Check the realm or organization name.

TABLE B–1 Log Reference for SAML v2 Plug-in for Federation Services *(Continued)*

Id	Description	Data	Trigger	Action
155	Configuration error thrown.	■ Error message ■ Realm or organization name	Retrieving all entities.	Check debug file for detailed information.
156	Successfully btained all entities.	Realm or organization name	Retrieving all entities.	

Index