Oracle® OpenSSO

Fedlet Interoperability Guide for Oracle Identity Federation

11*g* Release 1 (11.1.1.3.0)

E17847-01

July 2010

This guide explains how to deploy and configure the Oracle OpenSSO Fedlet with Oracle Identity Federation.

ORACLE

Oracle OpenSSO Fedlet Interoperability Guide for Oracle Identity Federation, 11*g* Release 1 (11.1.1.3.0)

E17847-01

Primary Author: John Spencer

Contributing Author: Vinaye Misra

Contents

Preface vii

Audience vii
Documentation Accessibility vii
Related Documents viii
Conventions viii

1 About the Oracle OpenSSO Fedlet

1.1 What is the Oracle OpenSSO Fedlet? 1-1
1.2 Oracle OpenSSO Fedlet Supported Standards and Applications 1-1
1.3 Installing and Configuring the Oracle OpenSSO Fedlet 1-2
1.4 Oracle OpenSSO Fedlet Features 1-3
1.4.1 Oracle OpenSSO Fedlet SAML 2.0 Single Sign-on (SSO) and Single Logout Features 1-3
1.4.2 Oracle OpenSSO Fedlet SAML 2.0 Discovery Service Features 1-3
1.4.3 Oracle OpenSSO Fedlet Additional SAML 2.0 Features 1-4
1.5 Oracle OpenSSO Fedlet Scenarios 1-4
1.5.1 New Oracle OpenSSO Fedlet Deployment 1-4
1.5.2 Oracle OpenSSO Fedlet Configuration Only 1-4
1.5.3 Oracle OpenSSO Fedlet SP-Initiated and IdP-Initiated SAML 2.0 Single Sign-on 1-5
1.5.4 Oracle OpenSSO Fedlet Single Logout 1-5
1.5.5 Oracle OpenSSO Fedlet Identity Provider Discovery Service with Multiple Identity Providers 1-5
1.5.6 Oracle OpenSSO Fedlet Signing and Encryption 1-5
1.5.7 Oracle OpenSSO Fedlet Attribute Query 1-5
1.5.8 Oracle Identity Federation as an Additional Identity Provider With OpenSSO 8.0 Update 1 1-6

2 Installing the Oracle OpenSSO Fedlet

2.1 Downloading the Oracle OpenSSO Fedlet ZIP File 2-1
2.2 Extracting the Oracle OpenSSO Fedlet Files 2-1
2.3 Packaging the Oracle OpenSSO Fedlet for a Service Provider 2-2

3 Configuring the Java Oracle OpenSSO Fedlet

3.1 Before You Configure the Java Oracle OpenSSO Fedlet 3-2
3.2 Configuring the Java Oracle OpenSSO Fedlet Using the `ConfigureFedlet` Program 3-3

3.2.1 Considerations for Running the `ConfigureFedlet` Program 3-3
3.2.1.1 Tasks Performed by the `ConfigureFedlet` Program 3-3
3.2.1.2 Considerations for Generating the `fedletsample.war` 3-4
3.2.2 Configuring the Java Oracle OpenSSO Fedlet Using the `ConfigureFedlet` Program . 3-4
3.2.3 Encrypting a Password Using the `ConfigureFedlet` Program 3-6
3.3 Configuring the Java Oracle OpenSSO Fedlet Manually 3-7
3.4 Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet 3-9
3.4.1 Generating the Metadata for an Oracle Identity Federation Identity Provider 3-9
3.4.2 Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet 3-10
3.5 Integrating the Java Oracle OpenSSO Fedlet into an Existing Application 3-11
3.6 Configuring the Java Oracle OpenSSO Fedlet with an Existing Application After Single Sign-On 3-12
3.7 Configuring the Java Oracle OpenSSO Fedlet for Single Logout 3-13
3.7.1 Implementing Single Logout for the Java Oracle OpenSSO Fedlet Service Provider Application 3-13
3.7.2 Implementing Single Logout for the Identity Provider 3-13
3.7.3 Implementing the Java Oracle OpenSSO Fedlet Adapter SPI for Single Logout 3-14
3.8 Configuring the Java Oracle OpenSSO Fedlet for Multiple Identity Providers 3-15
3.9 Configuring the Java Oracle OpenSSO Fedlet to Use the Identity Provider Discovery Service 3-16
3.10 Configuring the Java Oracle OpenSSO Fedlet for SAML 2.0 Attribute Query 3-17
3.11 Configuring the Java Oracle OpenSSO Fedlet for Signing and Encryption 3-19
3.12 Using the Oracle OpenSSO Fedlet Java APIs 3-23

4 Configuring the .NET Oracle OpenSSO Fedlet

4.1 Before You Configure the .NET Oracle OpenSSO Fedlet 4-2
4.2 Configuring the .NET Oracle OpenSSO Fedlet 4-2
4.3 Configuring Oracle Identity Federation as an Identity Provider for the .NET Oracle OpenSSO Fedlet 4-4
4.3.1 Generating the Metadata for an Oracle Identity Federation Identity Provider 4-4
4.3.2 Configuring an Oracle Identity Federation Identity Provider for the .NET Oracle OpenSSO Fedlet 4-5
4.4 Deploying the .NET Oracle OpenSSO Fedlet Sample Application 4-6
4.5 Configuring the .NET Oracle OpenSSO Fedlet with an Existing Application After Single Sign-on 4-6
4.6 Configuring the .NET Oracle OpenSSO Fedlet for Single Logout 4-8
4.7 Configuring the .NET Oracle OpenSSO Fedlet for Multiple Identity Providers 4-10
4.8 Configuring the .NET Oracle OpenSSO Fedlet to Use the Identity Provider Discovery Service 4-11
4.9 Configuring the .NET Oracle OpenSSO Fedlet for Signing of Requests and Responses 4-12
4.10 Configuring the .NET Oracle OpenSSO Fedlet for Encryption and Decryption of Requests and Responses 4-13

List of Tables

1–1 Oracle OpenSSO Fedlet SAML 2.0 Single Sign-on (SSO) and Single Logout Features ... 1-3
1–2 Oracle OpenSSO Fedlet SAML 2.0 Discovery Service Features ... 1-3
1–3 Oracle OpenSSO Fedlet Additional SAML 2.0 Features ... 1-4
2–1 Directory Structure of the `Oracle-OpenSSO-Fedlet.zip` File ... 2-2
3–1 Oracle OpenSSO Fedlet Java APIs ... 3-23

Preface

The Oracle OpenSSO Fedlet (Fedlet) is a lightweight service provider (SP) implementation that can be integrated with a Java or .NET application, enabling the application to communicate with an identity provider (IdP) such as an Oracle Identity Federation identity provider using the SAML 2.0 protocol.

Audience

This document is intended for the following users:

- Java or .NET developers who plan to integrate the Oracle OpenSSO Fedlet with a service provider (SP) application
- Administrators who plan to configure an identity provider such as an Oracle Identity Federation identity provider to use the Oracle OpenSSO Fedlet in a federated network environment

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

About the Oracle OpenSSO Fedlet

This chapter provides an introduction to the Oracle OpenSSO Fedlet, including:

- What is the Oracle OpenSSO Fedlet?
- Oracle OpenSSO Fedlet Supported Standards and Applications
- Installing and Configuring the Oracle OpenSSO Fedlet
- Oracle OpenSSO Fedlet Features
- Oracle OpenSSO Fedlet Scenarios

For information about federated identity management, including a description of the key features and concepts of Oracle Identity Federation, see the "Introduction to Oracle Identity Federation" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

1.1 What is the Oracle OpenSSO Fedlet?

The Oracle OpenSSO Fedlet (Fedlet) is a compact, easy to deploy SAML 2.0 service provider implementation. It includes a small software package and a simple file-based configuration, embeddable into a service provider's Java or .NET application. The Fedlet establishes single sign-on (SSO) between an identity provider instance and the service provider application without requiring a fully-featured federation product on the service provider side.

The Oracle OpenSSO Fedlet can accept SAML 2.0 assertions from any SAML 2.0 identity provider and retrieve user attributes to accomplish SSO and content personalization. The Fedlet can be configured to communicate with any number of identity providers. It also can leverage an external discovery service to find the preferred identity provider.

1.2 Oracle OpenSSO Fedlet Supported Standards and Applications

For information about the platforms and product versions supported by the Oracle OpenSSO Fedlet, see the appropriate certification matrix:

`http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html`

1.3 Installing and Configuring the Oracle OpenSSO Fedlet

The Oracle OpenSSO Fedlet can be downloaded as a separate ZIP file. The ZIP file includes all the files and components required to deploy the Fedlet with a Java or .NET service provider application. To use the Fedlet, you are not required to install any other federation components on the service provider side.

To install and configure the Oracle OpenSSO Fedlet, follow these general steps:

1. Download and unzip the `Oracle-OpenSSO-Fedlet.zip` file, as described in Chapter 2, "Installing the Oracle OpenSSO Fedlet."

   Note: For some deployments, rather than downloading the Oracle OpenSSO Fedlet ZIP file, a service provider administrator can simply get a previously configured Oracle OpenSSO Fedlet package from the identity provider administrator. The service provider administrator then adds any application specific logic to the package and deploys the Fedlet service provider application.

   To determine the Oracle OpenSSO Fedlet version, check the `FederationConfig.properties` file for the Java Fedlet or the `Fedlet.dll.config` file for the .NET Fedlet after you extract the files in the Fedlet package.

2. Get the metadata file from your identity provider, name this file `idp.xml`, and copy it to the Fedlet configuration directory.

3. Configure the Oracle OpenSSO Fedlet as follows:

   - Configure the Java Fedlet by running the `ConfigureFedlet` program or by performing the configuration steps manually, as described in Chapter 3, "Configuring the Java Oracle OpenSSO Fedlet."

     In most cases, you can run the `ConfigureFedlet` program, which prompts you for information and then automatically configures the Java Fedlet. If you want to configure specific features, however, such as Attribute Query, you must configure the Java Fedlet manually.

   - Configure the .NET Fedlet, as described in Chapter 4, "Configuring the .NET Oracle OpenSSO Fedlet."

4. A service provider administrator (or developer) can add any specific application logic to the service provider application. For example, for the Java Fedlet, add the logic to the `fedlet.war` or embed the `fedlet.war` in the service provider application.

5. Import the Fedlet service provider metadata file (`sp.xml`) into the identity provider. This file is created during the Fedlet configuration.

6. If you configured the Fedlet for features such as the identity provider discovery service or attribute query, perform the additional configuration steps on the identity provider side required for these features.

You can deploy multiple instances of the Oracle OpenSSO Fedlet on the same host as follows:

- Multiple Java Fedlet instances can run on the same host server if each Fedlet instance runs in its own Java Virtual Machine (JVM).
- Multiple .NET Fedlet instances can run on the same Internet Information Server (IIS), if you deploy the .NET Fedlet files in each respective application's `App_Data` and `bin` folders.

One consideration, however, is that the Oracle OpenSSO Fedlet does not perform session management on the service provider side. The service provider application or web container must perform the session management.

1.4 Oracle OpenSSO Fedlet Features

The Oracle OpenSSO Fedlet supports the following features:

- Oracle OpenSSO Fedlet SAML 2.0 Single Sign-on (SSO) and Single Logout Features
- Oracle OpenSSO Fedlet SAML 2.0 Discovery Service Features
- Oracle OpenSSO Fedlet Additional SAML 2.0 Features

1.4.1 Oracle OpenSSO Fedlet SAML 2.0 Single Sign-on (SSO) and Single Logout Features

Table 1–1 Oracle OpenSSO Fedlet ***SAML 2.0 Single Sign-on (SSO) and Single Logout Features***

Feature	Java Fedlet	.NET Fedlet
SAML 2.0 SSO		
IdP and SP Initiated HTTP POST	Yes	Yes
IdP and SP Initiated HTTP Artifact	Yes	Yes
SAML 2.0 Single Logout		
IdP and SP Initiated HTTP POST	Yes	Yes
IdP and SP Initiated HTTP Redirect	Yes	Yes

1.4.2 Oracle OpenSSO Fedlet SAML 2.0 Discovery Service Features

Table 1–2 Oracle OpenSSO Fedlet ***SAML 2.0 Discovery Service Features***

Feature	Java Fedlet	.NET Fedlet
Multiple IdP Support	Yes	Yes
External IdP Discovery Service	Yes	Yes
Bundled IdP Discovery Service (Reader Service Only)	Yes	No

1.4.3 Oracle OpenSSO Fedlet Additional SAML 2.0 Features

Table 1–3 Oracle OpenSSO Fedlet ***Additional SAML 2.0 Features***

Feature	Java Fedlet	.NET Fedlet
Signing of Requests and Response	Yes	Yes
Encryption of Attribute, Assertion, and NameID Elements	Yes	Yes
Export of SP Metadata	Yes	Yes
Attribute Query	Yes	No

1.5 Oracle OpenSSO Fedlet Scenarios

This section describes the following scenarios for the Oracle OpenSSO Fedlet:

- New Oracle OpenSSO Fedlet Deployment
- Oracle OpenSSO Fedlet Configuration Only
- Oracle OpenSSO Fedlet SP-Initiated and IdP-Initiated SAML 2.0 Single Sign-on
- Oracle OpenSSO Fedlet Single Logout
- Oracle OpenSSO Fedlet Identity Provider Discovery Service with Multiple Identity Providers
- Oracle OpenSSO Fedlet Signing and Encryption
- Oracle OpenSSO Fedlet Attribute Query
- Oracle Identity Federation as an Additional Identity Provider With OpenSSO 8.0 Update 1

1.5.1 New Oracle OpenSSO Fedlet Deployment

You want to download, install, and configure the Oracle OpenSSO Fedlet as new deployment on the service provider side in your environment. See these chapters:

- Chapter 2, "Installing the Oracle OpenSSO Fedlet"
- Chapter 3, "Configuring the Java Oracle OpenSSO Fedlet"
- Chapter 4, "Configuring the .NET Oracle OpenSSO Fedlet"

1.5.2 Oracle OpenSSO Fedlet Configuration Only

You have installed the Oracle OpenSSO Fedlet, and you want to configure or reconfigure your installation. See these chapters:

- Chapter 3, "Configuring the Java Oracle OpenSSO Fedlet"
- Chapter 4, "Configuring the .NET Oracle OpenSSO Fedlet"

1.5.3 Oracle OpenSSO Fedlet SP-Initiated and IdP-Initiated SAML 2.0 Single Sign-on

If you have installed the Oracle OpenSSO Fedlet, and you want to configure it for service provider initiated or identity provider initiated SAML 2.0 single sign-on (or both), see the following sections:

- Section 3.6, "Configuring the Java Oracle OpenSSO Fedlet with an Existing Application After Single Sign-On"
- Section 4.5, "Configuring the .NET Oracle OpenSSO Fedlet with an Existing Application After Single Sign-on"

1.5.4 Oracle OpenSSO Fedlet Single Logout

You have installed the Oracle OpenSSO Fedlet, and you want to configure single logout. Single logout allows the session termination of all participants in a session simultaneously. Any participant in the session can initiate the logout request. See the following sections:

- Section 3.7, "Configuring the Java Oracle OpenSSO Fedlet for Single Logout"
- Section 4.6, "Configuring the .NET Oracle OpenSSO Fedlet for Single Logout"

1.5.5 Oracle OpenSSO Fedlet Identity Provider Discovery Service with Multiple Identity Providers

Your existing identity federation deployment has the Oracle OpenSSO Fedlet configured with multiple identity providers in a circle of trust, and you want to configure the Oracle OpenSSO Fedlet to use the identity provider discovery service to determine the preferred identity provider. See the following sections:

- Section 3.9, "Configuring the Java Oracle OpenSSO Fedlet to Use the Identity Provider Discovery Service"
- Section 4.8, "Configuring the .NET Oracle OpenSSO Fedlet to Use the Identity Provider Discovery Service"

1.5.6 Oracle OpenSSO Fedlet Signing and Encryption

The Oracle OpenSSO Fedlet supports XML signature verification and decryption of encrypted `assertion` and `nameid` elements and their corresponding attributes. See the following sections:

- Section 3.11, "Configuring the Java Oracle OpenSSO Fedlet for Signing and Encryption"
- Section 4.9, "Configuring the .NET Oracle OpenSSO Fedlet for Signing of Requests and Responses"

1.5.7 Oracle OpenSSO Fedlet Attribute Query

You are a service provider that wants to use the Oracle OpenSSO Fedlet attribute query feature with an identity provider to retrieve user attributes to customize the service you provide for your users. See Section 3.10, "Configuring the Java Oracle OpenSSO Fedlet for SAML 2.0 Attribute Query." (The attribute query feature is not supported by the .NET Fedlet.)

1.5.8 Oracle Identity Federation as an Additional Identity Provider With OpenSSO 8.0 Update 1

Your existing identity federation deployment has the Oracle OpenSSO Fedlet installed with Oracle OpenSSO 8.0 Update 1 configured as an identity provider, and you want to add an Oracle Identity Federation identity provider to your deployment. See the following sections:

- Section 3.8, "Configuring the Java Oracle OpenSSO Fedlet for Multiple Identity Providers"
- Section 4.7, "Configuring the .NET Oracle OpenSSO Fedlet for Multiple Identity Providers"

2

Installing the Oracle OpenSSO Fedlet

This chapter explains how to install the Oracle OpenSSO Fedlet, including:

- Downloading the Oracle OpenSSO Fedlet ZIP File
- Extracting the Oracle OpenSSO Fedlet Files
- Packaging the Oracle OpenSSO Fedlet for a Service Provider

2.1 Downloading the Oracle OpenSSO Fedlet ZIP File

The Oracle OpenSSO Fedlet can be downloaded as a separate ZIP file. The ZIP file includes all the files and components required to deploy the Fedlet with a Java or .NET service provider application.

You can download the `Oracle-OpenSSO-Fedlet.zip` file from the Oracle Fusion Middleware 11gR1 Software Downloads page:

http://www.oracle.com/technology/software/products/middleware/htdocs/fmw_11_download.html

Create a directory to download and unzip the `Oracle-OpenSSO-Fedlet.zip` file. Usually, you should create this directory on the server where your service provider application exists. For example, on a UNIX system:

```
mkdir Fedlet-zip-dir
```

Note: For some deployments, rather than downloading the Oracle OpenSSO Fedlet ZIP file, a service provider administrator can get a previously configured Oracle OpenSSO Fedlet package from the identity provider administrator. The service provider administrator then adds any application specific logic to the package and deploys the Fedlet service provider application.

For more information, see Section 2.3, "Packaging the Oracle OpenSSO Fedlet for a Service Provider."

2.2 Extracting the Oracle OpenSSO Fedlet Files

In the download directory, extract the files in the `Oracle-OpenSSO-Fedlet.zip` file. For example:

```
cd Fedlet-zip-dir
unzip Oracle-OpenSSO-Fedlet.zip
```

The following table shows the directory structure of the `Oracle-OpenSSO-Fedlet.zip` file after you unzip the file. There are unique directories (and subdirectories) for the Java Fedlet and the .NET Fedlet.

Table 2–1 Directory Structure of the `Oracle-OpenSSO-Fedlet.zip` ***File***

Directory	Description
`java`	Files for integrating the Oracle OpenSSO Fedlet with a Java application: ■ `/conf` contains the Java Fedlet XML metadata templates, circle of trust template, and other configuration files. The `FederationConfig.properties` file contains the version number for the Oracle OpenSSO Fedlet release. ■ `fedlet.war` is a WAR file that you can deploy to show the Java Fedlet features. ■ `/SampleApp` contains a Java sample application that shows the connectivity between the Java Fedlet and a remote identity provider. ■ `/install` contains the `ConfigureFedlet` program to configure the Java Fedlet and optionally to create the Java Fedlet sample (`fedletsample.war`). ■ `README` provides information about the Java Fedlet.
`asp.net`	Files for integrating the Oracle OpenSSO Fedlet Fedlet with a .NET application: ■ `/bin` contains the `Fedlet.dll` and `Fedlet.dll.config` files. The `Fedlet.dll.config` file contains the version number for the Oracle OpenSSO Fedlet release. ■ `/conf` contains the .NET Fedlet XML metadata templates and circle of trust template. ■ `/SampleApp` contains a .NET sample application that shows the connectivity between the .NET Fedlet and a remote identity provider. ■ `readme.txt` provides information about the .NET Fedlet.

You are now ready to configure the Oracle OpenSSO Fedlet, as described in the following chapters:

- Chapter 3, "Configuring the Java Oracle OpenSSO Fedlet"
- Chapter 4, "Configuring the .NET Oracle OpenSSO Fedlet"

2.3 Packaging the Oracle OpenSSO Fedlet for a Service Provider

This section describes how an identity provider administrator can package the Oracle OpenSSO Fedlet with the identity provider metadata and send this package to a service provider administrator. The service provider administrator can then use this package to integrate a service provider Java or .NET application into the federated network environment.

To package the Oracle OpenSSO Fedlet with the identity provider metadata file, follow these steps:

1. On the identity provider side, download and unzip the Oracle OpenSSO Fedlet ZIP file, as described in Section 2.1, "Downloading the Oracle OpenSSO Fedlet ZIP File."
2. Get the Oracle OpenSSO Fedlet deployment URI from the service provider administrator.

3. Generate the identity provider metadata and save the metadata in a file named `idp.xml`.

   If you are using an Oracle Identity Federation identity provider, see Section 3.4, "Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet" or Section 4.3, "Configuring Oracle Identity Federation as an Identity Provider for the .NET Oracle OpenSSO Fedlet."

4. Configure the Oracle OpenSSO Fedlet, as described in Chapter 3, "Configuring the Java Oracle OpenSSO Fedlet" or Chapter 4, "Configuring the .NET Oracle OpenSSO Fedlet."

   During this configuration, use the deployment URI from Step 2 and the `idp.xml` file from Step 3.

5. Package the following items and give the package to the service provider administrator:

   - Java Fedlet: All files under the `java` directory and the output directory of your Java Fedlet configuration, including the `fedletsample.war` file if you generated this file during the configuration
   - .NET Fedlet: All files under the `asp.net` folder and your .NET Fedlet configuration files if they are not under the `asp.net` folder

   The service provider administrator (or developer) must add any necessary service provider application logic to the package and configure the service provider application for any additional features, such as using single logout or the identity provider discovery service.

6. Configure the identity provider by adding the Java Fedlet or .NET Fedlet as a trusted service provider and importing the service provider metadata (`sp.xml` file).

   If you are using an Oracle Identity Federation identity provider, see Section 3.4, "Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet" or Section 4.3, "Configuring Oracle Identity Federation as an Identity Provider for the .NET Oracle OpenSSO Fedlet."

   Note: The following additional configuration changes made to the Oracle OpenSSO Fedlet on the service provider side must be communicated to the identity provider administrator, so that the administrator can make the appropriate changes on the identity provider side:

   - Service provider metadata changes (`sp.xml` file) must be re-imported into the identity provider.
   - Service provider extended metadata changes (`sp-extended.xml` file) usually require configuration changes to the identity provider.

3

Configuring the Java Oracle OpenSSO Fedlet

This chapter describes how to configure the Java Oracle OpenSSO Fedlet (Java Fedlet) with a Java service provider (SP) application, so that the application can function with a remote identity provider (IdP) such as an Oracle Identity Federation (OIF) identity provider.

This chapter describes the following tasks for configuring the Java Oracle OpenSSO Fedlet:

- Before You Configure the Java Oracle OpenSSO Fedlet
- Configuring the Java Oracle OpenSSO Fedlet Using the `ConfigureFedlet` Program
- Configuring the Java Oracle OpenSSO Fedlet Manually
- Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet
- Integrating the Java Oracle OpenSSO Fedlet into an Existing Application
- Configuring the Java Oracle OpenSSO Fedlet with an Existing Application After Single Sign-On
- Configuring the Java Oracle OpenSSO Fedlet for Single Logout
- Configuring the Java Oracle OpenSSO Fedlet for Multiple Identity Providers
- Configuring the Java Oracle OpenSSO Fedlet to Use the Identity Provider Discovery Service
- Configuring the Java Oracle OpenSSO Fedlet for SAML 2.0 Attribute Query
- Configuring the Java Oracle OpenSSO Fedlet for Signing and Encryption
- Using the Oracle OpenSSO Fedlet Java APIs

Caution: When you configure the Oracle OpenSSO Fedlet, you might need to specify certain passwords, depending on the feature you are configuring. For example, for signing and encryption, you must provide passwords for the `.keypass` and `.storepass` files.

Although the Java Fedlet supports clear-text passwords, it is highly recommended in a production environment that you encrypt these passwords using the `fedletEncode.jsp` or as described in Section 3.2.3, "Encrypting a Password Using the `ConfigureFedlet` Program."

3.1 Before You Configure the Java Oracle OpenSSO Fedlet

Before you configure the Java Oracle OpenSSO Fedlet, perform these tasks:

- Download and unzip the `Oracle-OpenSSO-Fedlet.zip` file, as described in Chapter 2, "Installing the Oracle OpenSSO Fedlet."
- If you plan to deploy the Java Fedlet sample application, `fedlet.war`, or another Java application, deploy a supported web container, as listed in the Section 1.2, "Oracle OpenSSO Fedlet Supported Standards and Applications."
- If you plan to configure the Java Fedlet to use the attribute query feature and you are using an Oracle Identity Federation identity provider, set the Enable Attribute Query Responder option for the Oracle Identity Federation identity provider server instance before you generate the identity provider metadata (`idp.xml` file).

  For more information, see Section 3.4.1, "Generating the Metadata for an Oracle Identity Federation Identity Provider."

See Also: Chapter 1, "About the Oracle OpenSSO Fedlet."

Note: The tasks in this chapter include steps that must be performed on both the Java Fedlet service provider side and the identity provider side. Depending on your deployment, you might be performing all tasks yourself, or you might have to provide the Java Fedlet service provider information to an identity provider administrator. Several considerations are:

- The Java Fedlet service provider metadata file is named `sp.xml`.

  You or an identity provider administrator must import this file into the identity provider using the appropriate administration console or CLI command.

  If you reconfigure the Java Fedlet, and change the `sp.xml` file, re-import the revised file into the identity provider.

- The Java Fedlet service provider extended metadata file is named `sp-extended.xml`. Any changes you make in the file should be conveyed to the identity provider administrator, so that the appropriate changes can be made to the identity provider.

  For example, if you change the `wantLogoutResponseSigned` attribute to true, you should communicate this change to the administrator, so that the identity provider can be modified to sign the logout response.

3.2 Configuring the Java Oracle OpenSSO Fedlet Using the ConfigureFedlet Program

The ConfigureFedlet program configures the Java Oracle OpenSSO Fedlet without requiring you to perform some configuration steps manually. The ConfigureFedlet program can perform these functions:

- Configure the Java Fedlet and optionally generate the Java Fedlet sample application WAR file, as described in Section 3.2.2, "Configuring the Java Oracle OpenSSO Fedlet Using the ConfigureFedlet Program."
- Encrypt Fedlet passwords such as the keystore and private key passwords, as described in Section 3.2.3, "Encrypting a Password Using the ConfigureFedlet Program."

To determine whether you should run the ConfigureFedlet program or configure the Java Fedlet manually (or do both), see Section 3.2.1, "Considerations for Running the ConfigureFedlet Program."

After you configure and run the Java Fedlet, the debug logs are available under the Java Fedlet home /fedlet/debug directory. Log messages are written to the web container log files.

3.2.1 Considerations for Running the ConfigureFedlet Program

To configure the Java Fedlet, you can either run the ConfigureFedlet program or perform the configuration steps manually. You can also run the ConfigureFedlet program to do basic configuration for the Java Fedlet and then perform additional configuration manually, as required for your deployment. In some scenarios, such as configuring multiple identity providers or the identity provider discovery server, you must perform the manual configuration steps.

This section describes the following topics:

- Tasks Performed by the ConfigureFedlet Program
- Considerations for Generating the fedletsample.war

3.2.1.1 Tasks Performed by the ConfigureFedlet Program

This section describes the configuration tasks that the ConfigureFedlet can perform, so you can decide whether you want to run the program or perform the configuration steps manually.

The ConfigureFedlet program can perform the following configuration tasks for the Java Fedlet:

- Configure the Java Fedlet for single sign-on, single logout, and attribute query with the identity provider
- Generate the Java Fedlet service provider metadata
- Import the identity provider metadata
- Generate the default keystore and keypair for signing and encryption
- Generate the sample application WAR file

If you need to perform the following tasks, however, configure the Java Fedlet manually, as described in this chapter:

- Configure multiple identity providers
- Configure the identity provider discovery service

- Use a nondefault keystore and keypair
- Modify the extended identity provider metadata for customization, such as enabling signing for the logout request and response
- Integrate the Java Fedlet with a service provider application

3.2.1.2 Considerations for Generating the fedletsample.war

The ConfigureFedlet program asks whether you want to generate the fedletsample.war file during the configuration:

- If the fedletsample.war is generated, you can deploy the WAR file to a web container and use it as a sample Java Fedlet deployment. The WAR file contains the Java Fedlet source code, as well as samples, located in the SampleApp directory.
- If the fedletsample.war is not generated, you can bundle the fedlet.war in an EAR or WAR application file, customize the application if you wish, deploy the application, and then provide the configuration files generated by the ConfigureFedlet program.

3.2.2 Configuring the Java Oracle OpenSSO Fedlet Using the ConfigureFedlet Program

The ConfigureFedlet program configures the Java Fedlet by prompting you for information and then populating the Java Fedlet configuration files.

To determine whether you should run the ConfigureFedlet program or configure the Java Fedlet manually (or do both), see Section 3.2.1, "Considerations for Running the ConfigureFedlet Program."

When you run the ConfigureFedlet program, it displays the following prompts. You might want to obtain this information before you run the program:

- Enter the directory with path where Oracle-OpenSSO-Fedlet.zip is extracted to:
- Enter the URL where this Fedlet will be deployed on (in http(s)://host.domain:port/uri format):

  Note: If you are using Oracle Identity Federation as the identity provider and plan to configure the identity provider discovery service, https is required for the protocol.
- Enter Fedlet Provider ID:[fedlet_sp_sample]
- Do you want to generate keystore and key pair for the Fedlet? 1=yes/2=no [1]
- Enter Fedlet keystore password: and Re-enter Fedlet keystore password:
- Enter Fedlet key password: and Re-enter Fedlet key password:
- Do you want to import IDP metadata? 1=yes/2=no [1]
- Enter IDP metadata filename with path:
- Include sample and generate fedlet sample.war? 1=yes/2=no [1]
- Enter the directory with path where the newly generated Fedlet configuration and optionally fedletsample.war should be saved to:

To configure the Java Fedlet using the ConfigureFedlet program:

1. Make sure that $JAVA_HOME/bin is in your PATH variable, so that JDK commands such as jar, java, and keytool are accessible.

2. If necessary, download and unzip the Oracle-OpenSSO-Fedlet.zip, as described in Section 3.1, "Before You Configure the Java Oracle OpenSSO Fedlet."

   The ConfigureFedlet program is available in the Java Fedlet java/install directory after you unzip the Oracle-OpenSSO-Fedlet.zip file.

3. If you want the program to import the identity provider metadata, retrieve the metadata and store it in a file named idp.xml.

   For more information, see Section 3.4.1, "Generating the Metadata for an Oracle Identity Federation Identity Provider."

4. Extract the original fedlet.war file included in the Oracle-OpenSSO-Fedlet.zip. For example:

   ```
   cd WAR_DIR
   jar xvf FEDLET_ZIP_DIR/java/fedlet.war
   ```

5. Run the ConfigureFedlet program to configure and optionally to create the fedletsample.war.

   For example, on Solaris and Linux systems:

   ```
   java -classpath WAR_DIR/WEB-INF/lib/opensso-sharedlib.jar:
   WAR_DIR/WEB-INF/lib/openfedlib.jar:
   FEDLET_ZIP_DIR/java/install/lib/configurefedlet.jar
   oracle.security.fed.fedlet.install.ConfigureFedlet
   ```

   Or, on Windows systems:

   ```
   java -classpath WAR_DIR\WEB-INF\lib\opensso-sharedlib.jar;
   WAR_DIR\WEB-INF\lib\openfedlib.jar;
   FEDLET_ZIP_DIR\java\install\lib\configurefedlet.jar
   oracle.security.fed.fedlet.install.ConfigureFedlet
   ```

   The ConfigureFedlet program prompts you for input and then configures the Java Fedlet.

   The Java Fedlet configuration files are generated in the directory you specify.

   If you reply 1 (yes) to "Include sample and generate fedletsample.war?", the program also generates the fedletsample.war in the same location.

6. If you generated the fedletsample.war in the previous step, deploy the WAR to a web container using the Java Fedlet deployment URL you specified when you ran the program.

   If a fedletsample.war was not generated, copy FEDLET_OUTPUT_DIR/fedlet to a directory indicated by the system property user.home or com.sun.identity.fedlet.home.

   You can then add application logic to the fedlet.war or embed the fedlet.war in your application. Deploy the final WAR or EAR file to a web container to the Java Fedlet deployment URL you specified when you ran the ConfigureFedlet program.

7. The ConfigureFedlet program generates the service provider metadata in the sp.xml file in the Java Fedlet java/conf directory.

   Give the sp.xml file to the identity provider administrator to import into the identity provider.

Or, you can also give the Java Fedlet deployment URL to the identity provider administrator, so that the administrator can export the service provider metadata using the exportmetadata.jsp. For example:

```
https://fedlet-host.example.com:fedlet-port/uri/saml2/jsp/exportmetadata.jsp
```

By default, the Java Fedlet wants an assertion to be signed and signs its AuthnRequest (if the keystore is configured).

8. Configure the identity provider for the Java Fedlet by adding the Java Fedlet as a trusted service provider and importing the service provider metadata (sp.xml file).

   To configure an Oracle Identity Federation identity provider, see Section 3.4, "Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet."

9. The Java Fedlet is now configured and ready to run.

   If the fedletsample.war is deployed, you can test the Java Fedlet sample by accessing the Java Fedlet sample deployment URL. For example:

   ```
   https://fedlet-host.example.com:fedlet-port/uri/index.jsp
   ```

3.2.3 Encrypting a Password Using the ConfigureFedlet Program

You can run the ConfigureFedlet program to encrypt passwords such as the keystore and private key passwords. Use the encrypted passwords in the Java Fedlet configuration files for the Java Fedlet to use at run time.

For example, to encrypt a password:

1. Run the ConfigureFedlet program with the -e option. For example:

   ```
   java -classpath WAR_DIR/WEB-INF/lib/opensso-sharedlib.jar:
   WAR_DIR/WEB-INF/lib/openfedlib.jar:
   FEDLET_ZIP_DIR/java/install/lib/configurefedlet.jar
   -D"com.sun.identity.fedlet.home=fedlet-configuration-directory"
   oracle.security.fed.fedlet.install.ConfigureFedlet -e
   ```

2. Enter a plain text password at the following prompt:

   ```
   Enter password to be encrypted:
   ```

   The program encrypts and returns the password. For example:

   ```
   AQICVrMwQO5kkGifaGA88et605mWTFMPl18a
   ```

3. Copy the encrypted password to use with the Java Fedlet.

3.3 Configuring the Java Oracle OpenSSO Fedlet Manually

This section describes how to configure the Java Oracle OpenSSO Fedlet manually, without running the `ConfigureFedlet` program.

Note: If the `fedlet.war` or `fedletsample.war` files need to be customized to either include specific application logic or to integrate with an existing application, follow these steps:

1. Customize the `fedlet.war` application or include the `fedlet.war` in an EAR application file.
2. Perform the manual configuration steps in this section.
3. Deploy the WAR or EAR file.

To configure the Java Oracle OpenSSO Fedlet manually, follow these steps:

1. Create your Java Fedlet home directory, which is the directory where the Java Fedlet reads its metadata, circle of trust, and configuration properties files. The default location is the `fedlet` subdirectory under the home directory of the user running the Java Fedlet web container (indicated by the `user.home` JVM property).

   For example, if this home directory is `/home/webservd`, the Java Fedlet home directory is:

   ```
   /home/webservd/fedlet
   ```

   To change the Java Fedlet default home directory, set the value of the JVM run-time `com.sun.identity.fedlet.home` property to the desired location. For example:

   ```
   -Dcom.sun.identity.fedlet.home=/export/fedlet/conf
   ```

   The Java Fedlet then reads its metadata, circle of trust, and configuration files from the `/export/fedlet/conf` directory.

2. Copy the following configuration files from the Java Fedlet `java/conf` directory to the Java Fedlet home directory:
   - `sp.xml-template`
   - `sp-extended.xml-template`
   - `idp-extended.xml-template`
   - `fedlet.cot-template`
3. Rename the files you copied and drop `-template` from each name:
   - `sp-extended.xml`
   - `sp.xml`
   - `idp-extended.xml`
   - `fedlet.cot`

4. In the files that you copied and renamed in the Java Fedlet home directory, replace the tags shown in the next table:

Tag	Replace With
FEDLET_COT	Name of your circle of trust.
FEDLET_ENTITY_ID	ID (name) for the Java Fedlet service provider application. For example: `fedletsp`
FEDLET_PROTOCOL	Protocol of the web container for the Java Fedlet service provider application (such as `fedlet.war`). For example: `https` **Note**: If you are using an Oracle Identity Federation identity provider and plan to configure the identity provider discovery service, `https` is required for the protocol.
FEDLET_HOST	Host name of the web container for the Java Fedlet service provider application (such as `fedlet.war`). For example: `fedlet-host.example.com`
FEDLET_PORT	Port number of the web container for the Java Fedlet service provider application (such as `fedlet.war`). For example: 80
FEDLET_DEPLOY_URI	Deployment URI of the web container for the Java Fedlet service provider application (such as fedlet.war). For example: `fedlet`
IDP_ENTITY_ID	Entity ID (name) of the remote identity provider. For example: `oifidp`

   If the Java Fedlet service provider or identity provider entity ID contains a percent sign (%) or comma (,), escape the character before replacing it in the `fedlet.cot` file. For example, change "%" to "%25" and "," to "%2C".

5. Copy the `FederationConfig.properties` file from the Java Fedlet `java/conf` directory to the Java Fedlet home directory.

6. Get the identity provider standard metadata XML file and copy it to the Java Fedlet home directory. This file must be named `idp.xml`.

   For an Oracle Identity Federation identity provider, retrieve the metadata either from Oracle Enterprise Manager Fusion Middleware Control or by directly accessing a URL.

   Note: If you plan to configure the Java Fedlet to use the attribute query feature, set the Enable Attribute Query Responder option for the Oracle Identity Federation identity provider server instance before you generate the identity provider metadata (`idp.xml` file).

   For more information, see Section 3.4.1, "Generating the Metadata for an Oracle Identity Federation Identity Provider."

7. Configure the identity provider for the Java Fedlet by adding the Java Fedlet as a trusted service provider and importing the metadata file.

   To configure an Oracle Identity Federation identity provider, see Section 3.4, "Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet."

8. Test your Fedlet configuration for single sign-on and single logout follows:

   a. Deploy the `fedlet.war` file into your web container. Links to start the Java Fedlet service provider and identity provider initiated single sign-on are displayed.

   b. Click the links to be redirected to the identity provider for login and then for single sign-on to the Java Fedlet service provider.

      Upon successful completion, a Fedlet service provider JSP page shows the single sign-on response and assertion.

After you configure and run the Java Fedlet, the debug logs are available under the Java Fedlet home `/fedlet/debug` directory. Log messages are written to the web container log files.

3.4 Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet

This section describes how to configure an Oracle Identity Federation identity provider for the Java Oracle OpenSSO Fedlet, including:

- Generating the Metadata for an Oracle Identity Federation Identity Provider
- Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet

3.4.1 Generating the Metadata for an Oracle Identity Federation Identity Provider

To configure the Java Fedlet, you need the identity provider XML metadata in a file named `idp.xml`.

To generate the XML metadata for an Oracle Identity Federation identity provider, follow these steps:

1. Login to Oracle Fusion Middleware Control as an administrator who has the privileges required to manage the Oracle Identity Federation identity provider server instance you want to configure.
2. In Fusion Middleware Control, select the Oracle Identity Federation identity provider server instance in the topology panel at left.
3. If you are configuring the Java Fedlet for attribute query, set the Enable Attribute Query Responder option:

   a. Navigate to **Oracle Identity Federation**, **Administration**, and then **Identity Provider**.

   b. On the **SAML 2.0** tab, under **Protocol Settings**, check **Enable Attribute Query Responder**.

   c. Click **Apply**.

4. Generate the Oracle Identity Federation identity provider XML metadata, either from Oracle Enterprise Manager Fusion Middleware Control or by directly accessing a URL.

   To generate the Oracle Identity Federation identity provider metadata from Fusion Middleware Control:

   a. Navigate to **Oracle Identity Federation**, **Administration**, **Security and Trust**, and then **Provider Metadata**.

b. Select **Identity Provider** as the **Provider Type** and **SAML 2.0** as the **Protocol**.

c. Click **Generate**.

Or, to generate the Oracle Identity Federation identity provider metadata, go to a URL of the form:

```
http://host:port/fed/idp/metadata
```

5. Name the identity provider metadata file `idp.xml` and copy the file to the Java Fedlet home directory.

Continue with Section 3.2, "Configuring the Java Oracle OpenSSO Fedlet Using the `ConfigureFedlet` Program" or Section 3.3, "Configuring the Java Oracle OpenSSO Fedlet Manually."

3.4.2 Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet

After you configure the Java Oracle OpenSSO Fedlet on the service provider side and create the Java Fedlet service provider metadata file (`sp.xml`), configure the identity provider by adding the Java Fedlet as a trusted service provider and importing the metadata file.

To configure Oracle Identity Federation as an identity provider for the Java Fedlet, follow these steps:

1. Login to Oracle Fusion Middleware Control as an administrator who has the privileges required to manage the Oracle Identity Federation identity provider server instance you want to configure.
2. Add the Java Fedlet service provider as a trusted provider for the Oracle Identity Federation identity provider:
   a. In Fusion Middleware Control, select the Oracle Identity Federation identity provider server instance in the topology panel at left.
   b. Navigate to **Oracle Identity Federation**, **Administration**, and then **Federations**.
   c. On the **Federations** page, click **Add**.
   d. The **Add Trusted Provider** dialog appears. Upload the Java Fedlet service provider metadata file (`sp.xml`) from the file system.
   e. Click **OK**.
3. Configure the new Java Fedlet trusted provider you added in Step 2:
   a. Navigate to **Oracle Identity Federation**, **Administration**, and then **Federations**.
   b. On the **Federations** page, select the Java Fedlet trusted provider and click **Edit**.
   c. For **Oracle Identity Federation Settings**, select the following settings (if they have not been set globally):
      * Check **Enable Attributes in Single Sign-On (SSO)**.
      * If you are using the Java Fedlet attribute query feature, check **Send Signed** for **Response with Assertion - SOAP**.

Also, for the Java Fedlet attribute query feature, click **Edit** for **Attribute Mapping and Filters** and add the mappings for the attribute names. When you add an attribute pair, check **Get Value from User Session**.

d. Click **Apply**.

4. If you want to use identity provider initiated single sign-on and single logout, change the default **NameID** format to **Transient/One-Time Identifier** for the Java Fedlet service provider:

   a. In Fusion Middleware Control, select the Oracle Identity Federation identity provider server instance in the topology panel at left.

   b. Navigate to **Oracle Identity Federation**, **Administration**, and then **Federations**.

   c. On the **Federations** page, select the Java Fedlet trusted provider and click **Edit**.

   d. For **Oracle Identity Federation Settings**, set **Default NameID Format** to **Transient/One-Time Identifier**.

   e. Click **Apply**.

If you make additional configuration changes made to the Java Fedlet, such as changes made in the service provider extended metadata file (`sp-extended.xml`), convey these changes to the identity provider administrator, so that the appropriate changes can be made on the identity provider side.

If you reconfigure the Java Fedlet and change the `sp.xml` file, re-import the revised metadata file into the identity provider.

3.5 Integrating the Java Oracle OpenSSO Fedlet into an Existing Application

To integrate the Java Oracle OpenSSO Fedlet into an existing Java application, follow these steps:

1. If you have not configured the Java Fedlet, see Section 3.2, "Configuring the Java Oracle OpenSSO Fedlet Using the `ConfigureFedlet` Program" or Section 3.3, "Configuring the Java Oracle OpenSSO Fedlet Manually."

2. Extract the `fedlet.war` file in a temporary directory.

3. Copy all other files to your application WAR staging directory and overlay them with your existing application WAR structure.

   Optionally, you can remove `index.jsp`, `fedletEncode.jsp` and the Java Fedlet `java/conf` directory from the temporary directory created in Step 1 before you do the copy.

4. Create the new application WAR and redeploy it in your web container.

3.6 Configuring the Java Oracle OpenSSO Fedlet with an Existing Application After Single Sign-On

The `SampleApp` directory includes the sample Java Fedlet application named `fedletSampleApp.jsp`, which includes these functions and sample code:

- Invokes a util method to complete the SAML 2.0 protocol processing
- Returns a map containing various data, including response and assertion attributes for further processing by your application.
- Provides sample code showing how your application can retrieve the data from the returned map

You can either modify the `fedletSampleApp.jsp` and add your application specific application logic or replace the `fedletSampleApp.jsp` with your own servlet or JSP.

To replace the `fedletSampleApp.jsp` with a new servlet or JSP, follow these steps:

1. Modify your `web.xml` file to set the `servlet` and `servlet-mapping` elements for your new servlet or JSP. Map your new servlet or JSP to the `url-pattern` "`/fedletapplication`" because it is the URI set in the Java Fedlet metadata (the assertion consumer URL). For example:

```
<servlet>
 <servlet-name>yourapplication</servlet-name>
 <jsp-file>/Your-Application.jsp</jsp-file>
</servlet>
<servlet-mapping>
 <servlet-name>yourapplication</servlet-name>
 <url-pattern>/fedletapplication</url-pattern>
</servlet-mapping>
```

2. Copy following code from `fedletSampleApp.jsp` into your application using the appropriate import statement:

```
Map map;
try {
    // invoke the Fedlet processing logic. this will do all the
    // necessary processing conforming to SAMLv2 specifications,
    // such as XML signature validation, Audience and Recipient
    // validation etc.
    map = SPACSUtils.processResponseForFedlet(request, response);
} catch (SAML2Exception sme) {
    response.sendError(response.SC_INTERNAL_SERVER_ERROR, sme.getMessage());
    return;
} catch (IOException ioe) {
    response.sendError(response.SC_INTERNAL_SERVER_ERROR, ioe.getMessage());
    return;
} catch (SessionException se) {
    response.sendError(response.SC_INTERNAL_SERVER_ERROR, se.getMessage());
    return;
} catch (ServletException se) {
    response.sendError(response.SC_BAD_REQUEST, se.getMessage());
    return;
}
```

3. After your application obtains the returned `map` object, refer to the sample code in `fedletSampleApp.jsp` to retrieve the data required by your application.

3.7 Configuring the Java Oracle OpenSSO Fedlet for Single Logout

Single logout permits the session termination of all participants in a session simultaneously. Any participant in the session can initiate the logout request.

When using the Java Oracle OpenSSO Fedlet, the session state is maintained by the service provider application and not the Java Fedlet itself. Therefore, the service provider application performs the session termination and application logout for the user. The Java Fedlet simply provides support for the SAML 2.0 single logout communications. User log out from the application itself and single logout of the user from the identity provider can be initiated by the Java Fedlet or by the identity provider.

The Java Fedlet application decides whether to delete a user session before requesting single logout or after a successful single logout response is returned from the identity provider. It is also the Java Fedlet application's responsibility to invoke single logout for each identity provider with which it has successfully completed single sign on.

This section includes these topics about single logout:

- Implementing Single Logout for the Java Oracle OpenSSO Fedlet Service Provider Application
- Implementing Single Logout for the Identity Provider
- Implementing the Java Oracle OpenSSO Fedlet Adapter SPI for Single Logout

3.7.1 Implementing Single Logout for the Java Oracle OpenSSO Fedlet Service Provider Application

Single logout can be initiated on the Java Fedlet side using the HTTP Redirect profile or the HTTP POST profile. In the Java Fedlet initiated single logout, the user clicks on a link specified by the service provider application.

The Java Fedlet service provider application logs the user out locally, and then sends the logout information to the `spSingleLogoutInit.jsp`. The JSP forms a logout request and sends it to the identity provider. After receiving a successful logout response from the identity provider, the Java Fedlet notifies an `Adapter` class of the result of the Global Logout operation.

For that purpose, the Java Fedlet obtains an instance of the implementation of the `com.sun.identity.saml2.plugins.FedletAdapter` service provider interface (SPI), which contains the necessary methods for logout and post processing. The default implementation class, `com.sun.identity.saml2.plugins.DefaultFedletAdapter`, is provided with OpenSSO or you could write a new implementation. This section provides the following topics:

- Implementing Single Logout for the Java Oracle OpenSSO Fedlet Service Provider Application
- Implementing Single Logout for the Identity Provider

> **See Also:** Implementing the Java Oracle OpenSSO Fedlet Adapter SPI for Single Logout.

3.7.2 Implementing Single Logout for the Identity Provider

On the identity provider side, single logout can be initiated using the HTTP POST profile or the HTTP Redirect profile. In identity provider initiated single logout, the

Java Fedlet receives the logout request from the identity provider, verifies it, and obtains an instance of the `FedletAdapter` implementation.

The SPI invokes the `doFedletSLO()` method to log the user out of the service provider application. Then, the Java Fedlet forms a Logout Response based on the outcome and sends it to the identity provider. The default implementation of the Java Fedlet Adapter (`com.sun.identity.saml2.plugins.DefaultFedletAdapter`) is provided, or you can write a new implementation.

3.7.3 Implementing the Java Oracle OpenSSO Fedlet Adapter SPI for Single Logout

The `com.sun.identity.saml2.plugins.FedletAdapter` SPI provides the functionality for single logout, which can initiated by either the Java Fedlet service provider application or the identity provider. The `com.sun.identity.saml2.plugins.DefaultFedletAdapter` is the default implementation that is provided with the OpenSSO Fedlet.

- Single logout initiated by the Java Fedlet service provider application:

  For single logout initiated by the Java Fedlet service provider application, the `onFedletSLOSuccess()` and the `onFedletSLOFailure()` methods must be invoked after the Java Fedlet receives the logout response from the identity provider.

  If the SAML 2.0 single logout response is successful, the `onFedletSLOSuccess()` method is called. This method can perform post processing after a successful single logout. If you are using `com.sun.identity.saml2.plugins.DefaultFedletAdapter`, the method posts information regarding the logout to the `logout.jsp`, provided with the Java Fedlet.

  If the SAML 2.0 single logout response is not successful, the `onFedletSLOFailure()` method is called. This method is for post processing after an unsuccessful single logout and can perform developer-provided logic to handle the situation. If you are using `com.sun.identity.saml2.plugins.DefaultFedletAdapter`, it posts information regarding the logout to the `logout.jsp`, also provided with the Java Fedlet.

- Single logout initiated by the identity provider:

  For single logout initiated by the identity provider, the `doFedletSLO()` method must be invoked after the Java Fedlet receives the single logout request. The method initiates user log out from the application itself and the logout of the user from the identity provider.

You can write a new implementation by implementing the three methods as required in your application. Then, compile the class and integrate it into your application. If you are writing a new implementation, here are several considerations:

- Defining the `FedletAdapter` SPI Implementation in the Metadata

  The `fedletAdapterClass` attribute is new to the Java Fedlet's extended metadata. (It is not displayed in the entity provider extended metadata.) The value of `fedletAdapterClass` is the implementation of the `fedletAdapter` SPI, which by default is `com.sun.identity.saml2.plugins.DefaultFedletAdapter`.

- Using the `DefaultFedletAdapter` Implementation

`com.sun.identity.saml2.plugins.DefaultFedletAdapter` is the default implementation of the `FedletAdapter` provided with the Java Fedlet. It works in tandem with `logout.jsp`. Application specific logic related to the logout can be added to `logout.jsp` for functions such user management and session management. You can also use the `DefaultFedletAdapter` with your own JSP or servlet, but you must define the location of the page as the value of the `appLogoutUrl` attribute in the Java Fedlet extended metadata.

- Using the `logout.jsp`

  If you do not use the `logout.jsp` with your new implementation, you must write your own JSP or servlet and define it. See the previous item for more information.

3.8 Configuring the Java Oracle OpenSSO Fedlet for Multiple Identity Providers

In some deployments, you might want to configure the Java Oracle OpenSSO Fedlet with multiple identity providers. This section describes how to add one or more additional identity providers such as an Oracle Identity Federation identity provider.

This scenario also applies to an existing deployment with the Java Fedlet installed with Oracle OpenSSO 8.0 Update 1 as an identity provider, and you now want to add an additional Oracle Identity Federation identity provider. Oracle OpenSSO 8.0 Update 1 must be installed and running. For information about OpenSSO 8.0 Update 1, see the *OpenSSO 8.0 Update 1 Release Notes* at `http://docs.sun.com/doc/821-1818`.

Note: This section uses file names for adding a second identity provider. For example, `idp2.xml`, `fedlet2.cot`, and `idp2-extended.xml`. If necessary, rename these files using the appropriate number for the identity provider you are adding.

To configure the Java Fedlet for an additional identity provider such an Oracle Identity Federation identity provider, follow these steps:

1. Get the XML metadata file for the additional identity provider and name this file `idp2.xml`.

   For an Oracle Identity Federation identity provider, retrieve the metadata either from Oracle Enterprise Manager Fusion Middleware Control or by directly accessing a URL.

   To retrieve the Oracle Identity Federation identity provider metadata from Fusion Middleware Control:

   a. Navigate to **Oracle Identity Federation**, **Administration**, **Security and Trust**, and then **Provider Metadata**.

   b. Select **Identity Provider** as the **Provider Type** and **SAML 2.0** as the **Protocol**.

   c. Click **Generate**.

2. Add this new identity provider to the Java Fedlet circle of trust:

   To add the new identity provider to an existing circle of trust:

   In the `fedlet.cot` file in your Fedlet home directory, append the new identity provider entity ID (indicated by the `entityID` attribute in the `idp2.xml`

metadata file) to the value of the `sun-fm-trusted-providers` attribute, using a comma (,) as a separator.

To add the new identity provider to a new Java Fedlet circle of trust:

a. Create a new file named `fedlet2.cot` in your Fedlet home directory. Use the existing `fedlet.cot` as a template, but change the value of the `cot-name` attribute to the name of the new circle of trust (for example, `cot2`). Include both the new identity provider entity ID and the Java Fedlet entity ID as value for the `sun-fm-trusted-providers` attribute, with the two entity IDs separated by a comma (,).

b. In the `sp-extended.xml` file, add the new circle of trust name to the value of the cotlist attribute. For example, for a circle of trust named `cot2`:

```
<Attribute name="cotlist">
<Value>saml2cot</Value>
<Value>cot2</Value>
</Attribute>
```

3. In the Java Fedlet home directory, create a new `idp2-extended.xml` file as the extended metadata for the new identity provider. Use the existing `idp-extended.xml` file as a template, but change the entity ID to the new identity provider entity ID. Change the value for the `cotlist` attribute to the circle of trust name, if a new circle of trust is created for the identity provider.

   Make sure that the second identity provider is a remote identity provider by setting the hosted attribute in the `EntityConfig` element to `false`.

4. Restart the Java Fedlet web container.

5. The Java Fedlet metadata XML file (`sp.xml`) must be imported into the additional identity provider and added to the same circle of trust as the identity provider entity. Either import the `sp.xml` file into the identity provider, or give the file to your identity provider administrator to import.

   Note: Any additional configuration changes made to the Java Fedlet, such as changes made in the service provider extended metadata file (`sp-extended.xml`), must be communicated to the identity provider administrator, so that the appropriate changes can be made on the identity provider side.

   If you reconfigure the Java Fedlet and change the `sp.xml` file, re-import the revised file into the identity provider.

Repeat these steps for any additional identity providers you want to add for the Java Fedlet.

3.9 Configuring the Java Oracle OpenSSO Fedlet to Use the Identity Provider Discovery Service

If the Java Fedlet is configured with multiple identity providers in a circle of trust, you can configure the JAva Fedlet to use the identity provider discovery service to determine the preferred identity provider.

The discovery service must be configured for the identity providers you are using with the Java Fedlet. For information about configuring the identity provider discovery

service in Oracle Identity Federation, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

To configure the Java Fedlet to use the identity provider discovery service:

1. In the `FederationConfig.properties` file in the Java Fedlet home directory, set the following properties:

   ```
   com.iplanet.am.cookie.encode=true
   com.sun.identity.saml2.idpdiscovery.cookiedomain=cookie-domain
   ```

   where the *cookie-domain* follows the format `.example.com`

2. Restart the Java Fedlet web container.
3. Set up the identity provider discovery service for each of the remote identity providers.
4. Access the Java Fedlet `index.jsp` page, and you are presented with an identity provider selection page. Do not click the "use IDP discovery service..." link yet, because the preferred identity provider has not been set yet. Choose one of the identity providers, and complete the single sign-on process. The preferred identity provider is then set by the discovery service.
5. Check the cookies. You should find one `_saml_idp` cookie set.
6. Close and then reopen your browser. The `_saml_idp cookie` should be still be present.
7. Access the Java Fedlet `index.jsp` page again and choose the "use IDP discovery service to find out preferred IDP" link.

You are redirected to the identity provider discovery service to find the preferred identity provider and then sent to the Java Fedlet with the chosen identity provider to start the Java Fedlet initiated single sign-on.

3.10 Configuring the Java Oracle OpenSSO Fedlet for SAML 2.0 Attribute Query

The Java Oracle OpenSSO Fedlet supports SAML 2.0 attribute query to query an identity provider such as an Oracle Identity Federation identity provider for specific identity attribute values. You can configure the Java Fedlet to sign the query and encrypt the query. Signing is required for issuing a Java Fedlet query, but encryption is optional.

To configure the Java Fedlet for SAML 2.0 attribute query, follow these steps:

1. If you are using an Oracle Identity Federation identity provider, enable the Attribute Query Responder option for the Oracle Identity Federation identity provider server instance, as described in Section 3.4.1, "Generating the Metadata for an Oracle Identity Federation Identity Provider."
2. If you have not copied the identity provider standard metadata XML file (`idp.xml`) to the Java Fedlet home directory, copy this metadata file now.
3. Enable XML signing to sign the Attribute Query, as described in Section 3.11, "Configuring the Java Oracle OpenSSO Fedlet for Signing and Encryption." For Attribute Query, signing is required, but encryption is optional.
4. Add the certificate generated in the previous step to the `RoleDescriptor` element in the Java Fedlet `sp.xml` file.

If you run the `ConfigureFedlet` program to configure the Java Fedlet, you do not have to do this step manually. The program adds the certificate for you.

In the following example, there are two `KeyDescriptor` tags in which you paste the certificate. One is for signing and another is for encryption. If you are not enabling encryption, the `KeyDescriptor use="encryption"` tag is not required.

```
<RoleDescriptor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:query="urn:oasis:names:tc:SAML:metadata:ext:query"
      xsi:type="query:AttributeQueryDescriptorType"
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
   <KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:X509Data>
      <ds:X509Certificate>
                --certificate--
      </ds:X509Certificate>
      </ds:X509Data>
      </ds:KeyInfo>
   </KeyDescriptor>
   <KeyDescriptor use="encryption">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:X509Data>
      <ds:X509Certificate>
             --certificate--
      </ds:X509Certificate>
      </ds:X509Data>
      </ds:KeyInfo>
      <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
         <xenc:KeySize
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">128</xenc:KeySize>
      </EncryptionMethod>
   </KeyDescriptor>
</RoleDescriptor>
```

5. In the Java Fedlet `sp-extended.xml` file, specify the value for the `signingCertAlias` attribute and if configured, for the `encryptionCertAlias` attribute.

   If you run the `ConfigureFedlet` program to configure the Java Fedlet, you do not have to do this step manually.

   If you plan to configure the identity provider to encrypt the assertion, also encrypt the `NameID` element. Thus, the value of the `wantNameIDEncrypted` attribute must be set to true. Add the XML code to the `AttributeQueryConfig` element. For example:

   ```
   <Attribute name="signingCertAlias">
      <Value>test</Value>
   </Attribute>
   <Attribute name="encryptionCertAlias">
      <Value>test</Value>
   </Attribute>
   <Attribute name="wantNameIDEncrypted">
      <Value>true</Value>
   </Attribute>
   ```

   In this example, `test` is the alias for the sample key.

6. After you configure the Java Fedlet, update the identity provider with the changes you made to the Java Fedlet by importing the service provider metadata (`sp.xml`) into the identity provider.

   If you are using an Oracle Identity Federation identity provider, continue with Section 3.4.2, "Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet."

3.11 Configuring the Java Oracle OpenSSO Fedlet for Signing and Encryption

The Java Oracle OpenSSO Fedlet supports XML signature verification and decryption of encrypted `assertion` and `NameID` elements and their corresponding attributes. To configure the Java Fedlet for signing and encryption, follow these steps:

1. Create a keystore file named `keystore.jks` using the keytool utility.
2. Add the private key (and public certificate if applicable) used for signing and the private key (and public certificate if applicable) used for encryption to the `keystore.jks` file.
3. Create a `.storepass` file.
4. Add the password to the `.storepass` file. To encrypt the password, use `fedletEncode.jsp`. Or, see Section 3.2.3, "Encrypting a Password Using the `ConfigureFedlet` Program."
5. Create a `.keypass` file.
6. Add the password to the `.keypass` file. To encrypt the password, use `fedletEncode.jsp`. Or, see Section 3.2.3, "Encrypting a Password Using the `ConfigureFedlet` Program."
7. If you are using clear-text passwords (which is not recommended), comment out the following line in the `FederationConfig.properties` file:

   ```
   com.sun.identity.saml.xmlsig.passwordDecoder=
       com.sun.identity.fedlet.FedletEncodeDecode
   ```

8. Set the following attributes in the `FederationConfig.properties` file, where *path* is the complete path to the respective file:

   ```
   com.sun.identity.saml.xmlsig.keystore=path/keystore.jks
   com.sun.identity.saml.xmlsig.storepass=path/.storepass
   com.sun.identity.saml.xmlsig.keypass=path/.keypass
   ```

9. Use `keytool` to export the signing certificate. For example:

   ```
   keytool -export -keystore keystore.jks -rfc -alias test
   ```

   The tool prompts you to enter the password used to access `keystore.jks` and then generates the certificate.

10. If you need an encryption certificate, use `keytool` to export it, as shown in the previous step. (Or use the same certificate for both signing and encryption.)
11. Create a `KeyDescriptor` XML block and add the encryption certificate to it. Note the `use="signing"` tag of the `KeyDescriptor` element:

    ```
    <KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
    ```

```
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</KeyDescriptor>
```

12. Create another `KeyDescriptor` XML block and add the encryption certificate to it. Note the `use="encryption"` tag of the `KeyDescriptor` element:

```
<KeyDescriptor use="encryption">
  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
    <X509Data>
      <X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
      </X509Certificate>
    </X509Data>
  </KeyInfo>
<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
<KeySize xmlns="http://www.w3.org/2001/04/xmlenc#">128</KeySize>
</EncryptionMethod>
</KeyDescriptor>
```

13. In the `sp.xml` file, add the XML blocks with the signing and encryption certificates under the `SPSSODescriptor` element. For example:

```
<EntityDescriptor entityID="fedlet"
xmlns="urn:oasis:names:tc:SAML:2.0:metadata">
<SPSSODescriptor AuthnRequestsSigned="true" WantAssertionsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<b><KeyDescriptor use="signing">
  <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:X509Data>
      <ds:X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
```

```
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</KeyDescriptor></b>
<b><KeyDescriptor use="encryption">
  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
    <X509Data>
      <X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
      </X509Certificate>
    </X509Data>
  </KeyInfo>
<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
<KeySize xmlns="http://www.w3.org/2001/04/xmlenc#">128</KeySize>
</EncryptionMethod>
</KeyDescriptor></b>
<NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDFormat
>
<AssertionConsumerService index="1"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="http://server.sun.com:7070/fedlet/fedletapplication"/>
</SPSSODescriptor>
</EntityDescriptor>
```

The `AuthnRequestsSigned` attribute is set to true, configuring the Java Fedlet to sign all authentication requests.

14. In the Java Fedlet `sp-extended.xml` file, set values for the following attributes:
    - `signingCertAlias` contains the alias of the XML signing certificate in the keystore.
    - `encryptionCertAlias` contains the alias of the XML encryption certificate in the keystore.
15. To enforce what the Java Fedlet service provider encrypts, set the following attributes in the `sp-extended.xml` file to true:
    - `wantAssertionEncrypted`
    - `wantNameIDEncrypted`
    - `wantAttributeEncrypted`
16. To enforce what the Java Fedlet service provider signs and wants signed, set the following attributes to true.
    - `wantAuthnRequestsSigned` in the `idp.xml`file tells the Java Fedlet what to sign

- `AuthnRequestsSigned` and `WantAssertionsSigned` in the `sp.xml` file tells the identity provider what the Java Fedlet plans to sign.
- `wantArtifactResponseSigned` in the `sp-extended.xml` file tells the Java Fedlet what to sign.
- `wantPOSTResponseSigned` in the `sp-extended.xml` file
- `wantLogoutRequestSigned` in the `sp-extended.xml` file
- `wantLogoutResponseSigned` in the `sp-extended.xml` file

If the identity provider requires signing for specific messages, set the respective attributes to true in the `idp-extended.xml` file. For example, `wantLogoutRequestSigned` and `wantLogoutResponseSigned`.

Note: If you set attributes in the `sp-extended.xml` file, convey this information to the identity provider administrator, so that the necessary configuration changes can be made in the identity provider.

---

17. Restart the Java Fedlet web container.
18. Import the Java Fedlet `sp.xml` file into the identity provider.

    If you are using an Oracle Identity Federation identity provider, see Section 3.4.2, "Configuring an Oracle Identity Federation Identity Provider for the Java Oracle OpenSSO Fedlet."

3.12 Using the Oracle OpenSSO Fedlet Java APIs

The following table describes the Oracle OpenSSO Fedlet Java APIs that you can use in a service provider application.

See Also: *Oracle OpenSSO 8.0 Update 2 Java API Reference* in the following documentation collection:

`http://docs.sun.com/coll/1767.1`

The `getPolicyDecisionForFedlet` method described in the Java API reference is not currently supported.

Table 3–1 Oracle OpenSSO Fedlet Java APIs

Fedlet Java API	Description
`com.sun.identity.saml2.profile SPACSUtils processResponseForFedlet`	Processes responses from an identity provider to the Java Fedlet service provider application.
`com.sun.identity.saml2.profile AttributeQueryUtil getAttributeMapForFedlet`	Sends the attribute query to the specified attribute authority, validates the response, and returns the attribute map to the Java Fedlet.
`com.sun.identity.saml2.plugins FedletAdapter`	Provides the following methods: ▪ `initialize` initializes the Java Fedlet adapter (executed only once after the creation of the Java Fedlet Adapter instance). ▪ `doFedletSLO` logs out a user after the Java Fedlet service provider application receives a single logout request from an identity provider. ▪ `onFedletSLOSuccess` is invoked after the Java Fedlet service provider application receives a single logout response from the identity provider and the single logout status is a success. ▪ `onFedletSLOFailure` is invoked after the Java Fedlet service provider application receives a single logout response from the identity provider and the single logout status is not a success. For more information, see Section 3.7.3, "Implementing the Java Oracle OpenSSO Fedlet Adapter SPI for Single Logout."

4

Configuring the .NET Oracle OpenSSO Fedlet

This chapter describes how to configure the .NET Oracle OpenSSO Fedlet (.NET Fedlet) with a service provider (SP) application, so that the application can function with a remote identity provider (IdP) such as an Oracle Identity Federation (OIF) identity provider.

The .NET Oracle OpenSSO Fedlet includes the `Fedlet.dll`, which supports both identity provider and service provider initiated single sign-on (SSO) with POST and artifact binding. Multiple identity providers and the identity provider discovery service are also supported with single sign-on. In addition, both identity provider and service provider initiated single logout is supported. ASP.NET developers are provided with an API and an example sample application to retrieve an `AuthnResponse` from the identity provider.

This chapter describes the following tasks for configuring the .NET Oracle OpenSSO Fedlet:

- Configuring the .NET Oracle OpenSSO Fedlet
- Configuring Oracle Identity Federation as an Identity Provider for the .NET Oracle OpenSSO Fedlet
- Deploying the .NET Oracle OpenSSO Fedlet Sample Application
- Configuring the .NET Oracle OpenSSO Fedlet with an Existing Application After Single Sign-on
- Configuring the .NET Oracle OpenSSO Fedlet for Single Logout
- Configuring the .NET Oracle OpenSSO Fedlet for Multiple Identity Providers
- Configuring the .NET Oracle OpenSSO Fedlet to Use the Identity Provider Discovery Service
- Configuring the .NET Oracle OpenSSO Fedlet for Signing of Requests and Responses
- Configuring the .NET Oracle OpenSSO Fedlet for Encryption and Decryption of Requests and Responses

Caution: When you configure the Oracle OpenSSO Fedlet, you might need to specify certain passwords, depending on the feature you are configuring. For example, for signing and encryption, you must provide passwords for the `.keypass` and `.storepass` files.

Although the .NET Oracle OpenSSO Fedlet supports clear-text passwords, it is highly recommended in a production environment that you encrypt these passwords.

4.1 Before You Configure the .NET Oracle OpenSSO Fedlet

Before you configure the .NET Oracle OpenSSO Fedlet, download and unzip the `Oracle-OpenSSO-Fedlet.zip` file, as described in Chapter 2, "Installing the Oracle OpenSSO Fedlet."

For the supported components for the .NET Fedlet, see Section 1.2, "Oracle OpenSSO Fedlet Supported Standards and Applications."

For general information about the .NET Fedlet, see Chapter 1, "About the Oracle OpenSSO Fedlet."

Note: The tasks in this chapter include steps that must be performed on the .NET Oracle OpenSSO Fedlet service provider side and the identity provider side. Depending on your deployment, you might be performing all tasks yourself, or you might be required to provide the .NET Fedlet service provider information to an identity provider administrator. Several considerations are:

- The .NET Fedlet service provider metadata file is named `sp.xml`. You or an identity provider administrator must import this file into the identity provider using the appropriate administration console or CLI command.
- The .NET Fedlet service provider extended metadata file is named `sp-extended.xml`. Any changes made in the file must be communicated to the identity provider administrator, so that the appropriate changes can be made to the identity provider.

4.2 Configuring the .NET Oracle OpenSSO Fedlet

The .NET Oracle OpenSSO Fedlet does not include a configuration program. To configure the .NET Fedlet, perform these steps:

1. If you have not downloaded and unzipped the `Oracle-OpenSSO-Fedlet.zip` file, see Chapter 2, "Installing the Oracle OpenSSO Fedlet."
2. Copy the following files from the .NET Fedlet `asp.net/conf` folder to your application's `App_Data` folder:
   - `sp.xml-template`
   - `sp-extended.xml-template`
   - `idp-extended.xml-template`
   - `fedlet.cot-template`

3. Rename the files you copied, dropping `-template` from each name:
   - `sp-extended.xml`
   - `sp.xml`
   - `idp-extended.xml`
   - `fedlet.cot`
4. In the files you copied and renamed in the `App_Data` folder, replace the tags as shown in the next table:

Tag	Replace With
FEDLET_COT	Name of the circle of trust of which the remote identity provider and the local Fedlet service provider application are members.
FEDLET_ENTITY_ID	ID (name) for the .NET Fedlet service provider application. For example: `fedletsp`
FEDLET_DEPLOY_URI	URL of the .NET Fedlet service provider application. For example: `http://fedletsp.example.com/SampleApp`
IDP_ENTITY_ID	Entity ID (name) of the remote identity provider. For example: oifidp

   If the .NET Fedlet service provider or identity provider entity ID contains a percent sign (%) or comma (,), escape the character before replacing it in the `fedlet.cot` file. For example, change "%" to "%25" and "," to "%2C".
5. Get the identity provider metadata XML file and copy the file to the `App_Data` folder. This file must be named `idp.xml`.

   For an Oracle Identity Federation identity provider, see Section 4.3.1, "Generating the Metadata for an Oracle Identity Federation Identity Provider."
6. Copy the `Fedlet.dll` and the `Fedlet.dll.config` files from the .NET Fedlet `asp.net/bin` folder to your application's `bin` folder.
7. Configure the identity provider for the .NET Fedlet by adding the .NET Fedlet as a trusted service provider and importing the service provider metadata.

   To configure an Oracle Identity Federation identity provider, see Section 4.3, "Configuring Oracle Identity Federation as an Identity Provider for the .NET Oracle OpenSSO Fedlet."

   Tip: By deploying the .NET Fedlet artifacts in the application's `App_Data` and `bin` folders, you can deploy multiple instances of the .NET Fedlet in the same Internet Information Server (IIS), with each .NET Fedlet instance having its own files.

4.3 Configuring Oracle Identity Federation as an Identity Provider for the .NET Oracle OpenSSO Fedlet

Before you configure an Oracle Identity Federation identity provider, configure the .NET Oracle OpenSSO Fedlet on the service provider side and generate the service provider metadata file (`sp.xml`), as described in Section 4.2, "Configuring the .NET Oracle OpenSSO Fedlet."

This section includes the following information about configuring Oracle Identity Federation as an identity provider for the .NET Fedlet:

- Generating the Metadata for an Oracle Identity Federation Identity Provider
- Configuring an Oracle Identity Federation Identity Provider for the .NET Oracle OpenSSO Fedlet

4.3.1 Generating the Metadata for an Oracle Identity Federation Identity Provider

To configure the .NET Fedlet, generate the identity provider XML metadata and save the metadata in a file named `idp.xml`.

To generate the XML metadata for an Oracle Identity Federation identity provider, follow these steps:

1. Login to Oracle Fusion Middleware Control as an administrator who has the privileges required to manage the Oracle Identity Federation identity provider server instance you want to configure.
2. In Fusion Middleware Control, select the Oracle Identity Federation identity provider server instance in the topology panel at left.
3. Generate the Oracle Identity Federation identity provider XML metadata, either from Oracle Enterprise Manager Fusion Middleware Control or by directly accessing a URL.

   To generate the Oracle Identity Federation identity provider metadata from Fusion Middleware Control:

   a. Navigate to **Oracle Identity Federation**, **Administration**, **Security and Trust**, and then **Provider Metadata**.

   b. Select **Identity Provider** as the **Provider Type** and **SAML 2.0** as the **Protocol**.

   c. Click **Generate**.

   Or, to generate the Oracle Identity Federation identity provider metadata, go to a URL of the form:

   ```
   http://host:port/fed/idp/metadata
   ```

4. Name the identity provider metadata file `idp.xml` and copy the file to the .NET Fedlet `App_Data` folder.

Continue with Section 4.2, "Configuring the .NET Oracle OpenSSO Fedlet."

4.3.2 Configuring an Oracle Identity Federation Identity Provider for the .NET Oracle OpenSSO Fedlet

After you configure the .NET Oracle OpenSSO Fedlet on the service provider side and create the .NET Fedlet service provider metadata file (`sp.xml`), configure the identity provider by adding the .NET Fedlet as a trusted service provider and importing the metadata file.

To configure Oracle Identity Federation as an identity provider for the .NET Oracle OpenSSO Fedlet, follow these steps:

1. Login to Oracle Fusion Middleware Control as an administrator who has the privileges required to manage the Oracle Identity Federation identity provider server instance you want to configure.
2. Add the .NET Fedlet service provider as a trusted provider for the Oracle Identity Federation identity provider:
   a. In Fusion Middleware Control, select the Oracle Identity Federation identity provider server instance in the topology panel at left.
   b. Navigate to **Oracle Identity Federation**, **Administration**, and then **Federations**.
   c. On the **Federations** page, click **Add**.
   d. The **Add Trusted Provider** dialog appears. Upload the .NET Fedlet service provider metadata file (`sp.xml`) from the file system.
   e. Click **OK**.
3. Configure the new .NET Fedlet trusted provider you added in Step 2:
   a. Navigate to **Oracle Identity Federation**, **Administration**, and then **Federations**.
   b. On the **Federations** page, select the .NET Fedlet trusted provider and click **Edit**.
   c. For **Oracle Identity Federation Settings**, check **Enable Attributes in Single Sign-On (SSO)**.
   d. Click **Apply**.
4. If you want to use identity provider initiated single sign-on and single logout, change the default **NameID** format to **Transient/One-Time Identifier** for the .NET Fedlet service provider:
   a. In Fusion Middleware Control, select the Oracle Identity Federation identity provider server instance in the topology panel at left.
   b. Navigate to **Oracle Identity Federation**, **Administration**, and then **Federations**.
   c. On the **Federations** page, select the .NET Fedlet trusted provider and click **Edit**.
   d. For **Oracle Identity Federation Settings**, set **Default NameID Format** to **Transient/One-Time Identifier**.
   e. Click **Apply**.

If you make additional configuration changes made to the .NET Oracle OpenSSO Fedlet, such as changes made in the SP extended metadata file (`sp-extended.xml`), convey these changes to the identity provider administrator, so that the appropriate

changes can be made on the identity provider side. If you reconfigure the .NET Fedlet and change the `sp.xml` file, re-import the revised file into the identity provider.

4.4 Deploying the .NET Oracle OpenSSO Fedlet Sample Application

The .NET Oracle OpenSSO Fedlet sample application is available in the .NET Fedlet `asp.net/SampleApp` folder after you unzip the `Oracle-OpenSSO-Fedlet.zip` file. You can use this sample application to test your deployment of the .NET Fedlet for a .NET application.

To deploy the .NET Oracle OpenSSO Fedlet sample application, follow these steps:

1. If you have not configured the .NET Fedlet, follow the steps in Section 4.2, "Configuring the .NET Oracle OpenSSO Fedlet."
2. Copy the following edited files to the .NET Fedlet `asp.net/SampleApp/App_Data` folder:
   - `idp.xml`
   - `idp-extended.xml`
   - `sp.xml`
   - `sp-extended.xml`
   - `fedlet.cot`
3. Within Internet Information Server, create a virtual directory with the `SampleApp` folder within the unzipped folder.
   - In IIS 6, use Add Virtual Directory. Be sure to have Read and Script permissions set for the application.
   - In IIS 7, use Add Application (with no additional options required).

Now, you are ready to run the sample application:

1. Open the sample application in your browser. For example:

   ```
   http://fedletsp.example.com/SampleApp
   ```

2. Click the link to perform the identity provider initiated single sign-on.
3. Enter your credentials.
4. After the form submission, you should be at the `fedletapplication.aspx` page with access to the `AuthnResponse` information.

4.5 Configuring the .NET Oracle OpenSSO Fedlet with an Existing Application After Single Sign-on

The .NET Oracle OpenSSO Fedlet supports service provider initiated and identity provider initiated single sign-on.

To see how the single sign-on feature works, consider deploying the .NET Fedlet sample application, as described in Section 4.4, "Deploying the .NET Oracle OpenSSO Fedlet Sample Application."

To configure the .NET Oracle OpenSSO Fedlet with an existing application for single sign-on, follow these steps:

1. If you have not configured the .NET Fedlet, follow the steps in Section 4.2, "Configuring the .NET Oracle OpenSSO Fedlet."
2. After you configure the .NET Fedlet, copy the following configured files to the respective folders for your application:
   - Your application's `App_Data` folder:
     - `sp.xml`
     - `sp-extended.xml`
     - `idp.xml` (metadata from the identity provider)
     - `idp-extended.xml`
     - `fedlet.cot`
   - Your application's `bin` folder:
     - `Fedlet.dll`
     - `Fedlet.dll.config`
3. If you have not imported your application's metadata (`sp.xml` file) into the identity provider, import this metadata file now.

   If you are using an Oracle Identity Federation identity provider, see Section 4.3.2, "Configuring an Oracle Identity Federation Identity Provider for the .NET Oracle OpenSSO Fedlet."
4. Within your .NET application's public content folder, create a file based on the `spinitiatedsso.aspx` file in the .NET Fedlet `asp.net/SampleApp` folder.

   The `spinitiatedsso.aspx` file is used to trigger a service provider initiated single sign-on operation. For a list of supported query parameters, view the contents of this file. The supported query parameters are included as comments at the beginning of the file.
5. In your application, create the required links, depending on the features you are using. The following example links use *http://idp-host.example.com:port/uri* as the identity provider URL and *http://fedletsp.example.com/MyApp* for your application's URL.
   - Service provider initiated single sign-on using HTTP POST binding:

     ```
     spinitiatedsso.aspx?idpEntityId=http://idp-host.example.com:port/uri
     &;binding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST
     ```

   - Service provider initiated single sign-on using HTTP artifact binding:

     ```
     spinitiatedsso.aspx?idpEntityId=http://idp-host.example.com:port/uri
     &;binding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact
     ```

   - Identity provider initiated single sign-on using HTTP POST binding:

     ```
     http://idp-host.example.com:port/uri/idpssoinit
     ?NameIDFormat=urn:oasis:names:tc:SAML:2.0:nameid-format:transient
     &;metaAlias=/idp&;spEntityID=http://fedletsp.example.com/MyApp
     &;binding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST
     ```

   - Identity provider initiated single sign-on using HTTP artifact binding:

     ```
     http://idp-host.example.com:port/uri/idpssoinit
     ```

```
?NameIDFormat=urn:oasis:names:tc:SAML:2.0:nameid-format:transient
&;metaAlias=/idp&;spEntityID=http://fedletsp.example.com/MyApp
&;binding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact
```

6. After your application is successfully authenticated by the identity provider, your browser is redirected to your application's page that processes the SAML 2.0 response.

   Your application can then consume the SAML 2.0 response by modifying and using this code:

```
AuthnResponse authnResponse = null;
try
{
    ServiceProviderUtility spu = new ServiceProviderUtility(Context);
    authnResponse = spu.GetAuthnResponse(Context);
}
catch (Saml2Exception se)
{
    // invalid AuthnResponse received
}
catch (ServiceProviderUtilityException spue)
{
    // issues with deployment (reading metadata)
}
```

   In the SAML2.0 response, the `authnResponse` object is populated with the assertion information. Your application can then access the attributes from the response.

   If you deployed the sample application, it shows how to retrieve the attributes and subject information from this object.

4.6 Configuring the .NET Oracle OpenSSO Fedlet for Single Logout

Single logout permits the session termination of all participants in a session simultaneously. Any participant in the session can initiate the logout request. The .NET Oracle OpenSSO Fedlet supports both identity provider initiated and service provider initiated single logout.

When using the .NET Fedlet, the session state is maintained by the service provider application and not the .NET Fedlet itself. Therefore, the service provider application performs the session termination and application logout for a user. The .NET Fedlet simply provides the support for the SAML 2.0 single logout communications.

For .NET Fedlet initiated single logout, the .NET Fedlet application logs the user out locally and then invokes the .NET Fedlet for the global logout.

The .NET Fedlet application decides whether to delete a user session before requesting single logout or after a successful single logout response is returned from the identity provider. The .NET Fedlet application also has the responsibility to invoke single logout for each identity provider with which it has successfully completed single sign-on.

To implement single logout, the .NET Fedlet sample application includes the `logout.aspx` and `spinitiatedslo.aspx` files in the `asp.net/SampleApp` folder. To see how the single logout feature works, consider deploying the .NET Fedlet sample application, as described in Section 4.4, "Deploying the .NET Oracle OpenSSO Fedlet Sample Application."

To configure a .NET Oracle OpenSSO Fedlet service provider application for single logout, follow these steps:

1. If you have not configured the .NET Fedlet, follow the steps in Section 4.2, "Configuring the .NET Oracle OpenSSO Fedlet."
2. After you configure the .NET Fedlet, copy the following configured files to the respective folders for your application:
   - Your application's `App_Data` folder:
     - `sp.xml`
     - `sp-extended.xml`
     - `idp.xml` (metadata from the identity provider)
     - `idp-extended.xml`
     - `fedlet.cot`
   - Your application's `bin` folder:
     - `Fedlet.dll`
     - `Fedlet.dll.config`
3. If you have not imported your application's metadata (`sp.xml` file) into the identity provider, import this metadata now.

   If you are using an Oracle Identity Federation identity provider, see Section 4.3.2, "Configuring an Oracle Identity Federation Identity Provider for the .NET Oracle OpenSSO Fedlet."
4. Within your .NET application's public content folder, create files based on the `logout.aspx` and `spinitiatedslo.aspx` files in the .NET Fedlet `asp.net/SampleApp` folder.

   These files are used by the .NET Fedlet sample application to implement single logout. The `logout.aspx` file is used for identity provider initiated single logout, and the `spinitiatedslo.aspx` file is used for service provider initiated single logout. You can use them to develop your .NET Fedlet application.
5. Make these changes to the configuration files for your application:
   - For identity provider initiated single logout: In the `sp.xml` file, make sure the path to your file based on `logout.aspx` file points to the correct location of the file for your application.
   - For service provider initiated single logout: In the `idp.xml` file, make sure the path to your file based on `spinitiatedslo.aspx` file points to the correct location of the file for your application.
6. If you want the logout request and logout response signed, set the following attributes to `true` in the `sp-extended.xml` and `idp-extended.xml` files:
   - `wantLogoutRequestSigned`
   - `wantLogoutResponseSigned`
7. Import the .NET Fedlet service provider metadata file (`sp.xml`) into the identity provider. (If you have already imported the `sp.xml` file into the identity provider, re-import the updated file.)

   For an Oracle Identity Federation identity provider, see Section 4.3.2, "Configuring an Oracle Identity Federation Identity Provider for the .NET Oracle OpenSSO Fedlet."

Also, inform the identity provider administrator that you configured single logout for the .NET Fedlet service provider, so that the administrator can make any required additional changes to the identity provider configuration.

4.7 Configuring the .NET Oracle OpenSSO Fedlet for Multiple Identity Providers

In some deployments, you might want to configure the .NET Oracle OpenSSO Fedlet with multiple identity providers. This section describes how to add one or more additional identity providers such as an Oracle Identity Federation identity provider.

This use case also applies to an existing deployment with the .NET Fedlet installed with Oracle OpenSSO 8.0 Update 1 as the identity provider, and you now want to add Oracle Identity Federation as an additional identity provider. OpenSSO 8.0 Update 1 must be installed and running. For information about OpenSSO Update 1, see the *OpenSSO 8.0 Update 1 Release Notes* at `http://docs.sun.com/doc/821-1818`.

To configure the .NET Oracle OpenSSO Fedlet for an additional identity provider such as an Oracle Identity Federation identity provider, follow these steps:

1. Get the XML metadata file for the additional identity provider.

   You can retrieve the Oracle Identity Federation identity provider metadata either from Oracle Enterprise Manager Fusion Middleware Control or by directly accessing a URL.

   To retrieve the Oracle Identity Federation identity provider metadata from Fusion Middleware Control:

   a. Navigate to Oracle Identity Federation, Administration, Security and Trust, and then Provider Metadata.

   b. Select Identity Provider as the Provider Type and SAML 2.0 as the Protocol.

   c. Click Generate.

   Or, to get the Oracle Identity Federation identity provider metadata, go to a URL of the form:

   ```
   http://host:port/fed/idp/metadata
   ```

2. Name the additional identity provider metadata file as `idpn.xml`, where `n` is the Oracle Identity Federation identity provider that you are adding. For example, name the second identity provider file as `idp2.xml`, the third as `idp3.xml`, and so on. This procedure uses `idp2.xml` as the file name.

3. Copy the `idp2.xml` file from Step 2 to your application's `App_Data` folder.

4. Add this new identity provider to the .NET Fedlet circle of trust:

   To add the new identity provider to an existing circle of trust:

   In the `fedlet.cot` file in your application's `App_Data` folder, append the new IDP entity ID (indicated by the `entityID` attribute in the `idp2.xml` metadata file) to the value of the `sun-fm-trusted-providers` attribute, using a comma (,) as a separator.

   To add the new identity provider to a new circle of trust:

   a. Create a new file named `fedlet2.cot` in your application's `App_Data` folder. Use the existing fedlet.cot as a template, but change the value of the `cot-name` attribute to the name of the new circle of trust (for example, `cot2`). Include both the new identity provider entity ID and the .NET Fedlet entity ID

as value for the `sun-fm-trusted-providers` attribute, with the two entity IDs separated by a comma (,).

b. In the `sp-extended.xml` file, add the new circle of trust name to the value of the cotlist attribute. For example, for a circle of trust named `cot2`:

```
<Attribute name="cotlist">
<Value>saml2cot</Value>
<Value>cot2</Value>
</Attribute>
```

5. In your application's `App_Data` folder, create a new `idp2-extended.xml` file as the extended metadata for the new identity provider. Use the existing `idp-extended.xml` file as a template, but change the entityID to the new identity provider entity ID. Change the value for the `cotlist` attribute to the circle of trust name, if a new circle of trust is created for the identity provider.

   Note: Make sure that the second identity provider is a remote identity provider by setting the hosted attribute in the `EntityConfig` element to false.

6. Restart the Application Pool associated with your Fedlet .NET application.

7. Import the .NET Fedlet metadata XML file (`sp.xml`) into the additional identity provider and add the .NET Fedlet service provider to the same circle of trust as the identity provider.

   For information about using an Oracle Identity Federation identity provider, see Section 4.3.2, "Configuring an Oracle Identity Federation Identity Provider for the .NET Oracle OpenSSO Fedlet."

Repeat these steps for any additional identity providers you want to add.

4.8 Configuring the .NET Oracle OpenSSO Fedlet to Use the Identity Provider Discovery Service

In this scenario, the .NET Oracle OpenSSO Fedlet is configured with multiple identity providers in a circle of trust and you want to configure the .NET Fedlet to use the identity provider discovery service to determine the preferred identity provider.

The discovery service must be configured for the identity providers you are using with the .NET Fedlet. For information about configuring the identity provider discovery service in Oracle Identity Federation, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

To configure the .NET Oracle OpenSSO Fedlet to use the identity provider discovery service:

1. In the .NET Fedlet `fedlet.cot` file, set the `sun-fm-saml2-readerservice-url` property to the URL for the SAML 2.0 reader service. For example:

   ```
   sun-fm-saml2-readerservice-url=http://discovery.common.com/opensso/saml2reader
   ```

2. Restart the Application Pool associated with your .NET Fedlet application.

4.9 Configuring the .NET Oracle OpenSSO Fedlet for Signing of Requests and Responses

To configure the .NET Oracle OpenSSO Fedlet for signing of requests and responses:

1. Import your X.509 certificate to the Personal folder within the Local Computer account using the Certificates Snap-in for the Microsoft Management Console. To use this snap-in, see the following article:

   http://msdn.microsoft.com/en-us/library/ms788967.aspx

2. Specify a friendly name for this certificate by viewing the Properties dialog and entering a value. (Save this value for Step 4.)

3. Set the appropriate permissions to allow read access to the certificate for the user account used by Internet Information Server (IIS) as described at the Microsoft article. For example:

   a. In the Certificates Snap-in, navigate to Action, All Tasks, and then Manage Private Keys.

   b. Specify Allow Read permissions for the user account running IIS (usually NETWORK SERVICE).

4. In the .NET Fedlet's extended metadata file (`sp-extended.xml`), specify the friendly name specified in Step 2 as the value for the `signingCertAlias` attribute. For example:

```
<Attribute name="signingCertAlias">
<Value>MyFedlet</Value>
```

5. In the .NET Fedlet's service provider metadata file (`sp.xml`), add the `KeyDescriptor` for the signing key.

   Use the Certificates Snap-in for the Microsoft Management Console used earlier to export the public key of your certificate in Base64 encoding to be included in the `KeyDescriptor` XML block.

   This `KeyDescriptor` must be the first child element within the `SPSSODescriptor`. For example:

```
<KeyDescriptor use="signing">
              <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                  <ds:X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
                  </ds:X509Certificate>
                </ds:X509Data>
              </ds:KeyInfo>
</KeyDescriptor>
```

6. Restart the Application Pool associated with your .NET application.

4.10 Configuring the .NET Oracle OpenSSO Fedlet for Encryption and Decryption of Requests and Responses

To configure the .NET Oracle OpenSSO Fedlet for encryption and decryption of requests and responses:

1. Import your X.509 certificate to the Personal folder within the Local Computer account using the Certificates Snap-in for the Microsoft Management Console. To use this snap-in, see the following article:

   http://msdn.microsoft.com/en-us/library/ms788967.aspx

2. Specify a friendly name for this certificate by viewing the Properties dialog and entering a value. (Save this value for Step 4.)

3. Set the appropriate permissions to allow read access to the certificate for the user account used by Internet Information Server (IIS) as described at the Microsoft article. For example:

   a. In the Certificates Snap-in, navigate to Action, All Tasks, and then Manage Private Keys.

   b. Specify Allow Read permissions for the user account running IIS (usually NETWORK SERVICE).

4. In the .NET Fedlet's extended metadata file (`sp-extended.xml`), specify the friendly name specified in Step 2 as the value for the `encryptionCertAlias` attribute. For example:

```
<Attribute name="encryptionCertAlias">
<Value>MyFedlet</Value>
```

5. In the .NET Fedlet's service provider metadata file (`sp.xml`), add the `KeyDescriptor` for the encryption key.

   Use the Certificates Snap-in for the Microsoft Management Console used earlier to export the public key of your certificate in Base64 encoding to be included in the `KeyDescriptor` XML block.

   This `KeyDescriptor` must be the first child element within the `SPSSODescriptor`. For example:

```
<KeyDescriptor use="encryption">
              <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                  <ds:X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
                  </ds:X509Certificate>
                </ds:X509Data>
              </ds:KeyInfo>
              <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
                <KeySize
```

```
xmlns="http://www.w3.org/2001/04/xmlenc#">128</KeySize>
            </EncryptionMethod>
</KeyDescriptor>
```

6. Restart the Application Pool associated with your .NET Fedlet application.

To test this configuration, use the sample application. In addition, the following attributes can be changed to encrypt requests and decrypt responses with the identity provider with the appropriate changes to the configured metadata:

- Assertion: In the `sp-extended.xml` metadata file, set the `wantAssertionEncrypted` attribute to `true`, to have the .NET Fedlet decrypt the `EncryptedAssertion` element in incoming responses from the identity provider.
- Attribute: In the `sp-extended.xml` metadata file, set the `wantAttributeEncrypted` attribute to `true` to have the .NET Fedlet decrypt the `EncryptedAttribute` element in incoming responses from the identity provider.
- NameID: In the `idp-extended.xml` metadata file, set the `wantNameIDEncrypted` attribute to `true` to have the .NET Fedlet encrypt the `NameID` element in outgoing requests. Set this same attribute in `sp-extended.xml` to have the .NET Fedlet decrypt the `EncryptedID` element in incoming responses from the identity provider.