Oracle® Fusion Middleware

Administering Oracle Identity Federation

11*g* Release 1 (11.1.1)

E55733-01

April 2015

Describes how to deploy and manage Oracle Identity Federation, including how to configure to operate with federated partners.

ORACLE®

Oracle Fusion Middleware Administering Oracle Identity Federation, 11*g* Release 1 (11.1.1)

E55733-01

Primary Author: Vinaye Misra

Contributors: Deepthi Buddhiraju, Damien Carru, Mrudula Gaidhani, Ari Kermaier, Maya Neelakandhan, Naveen Kumar Vemula

Contents

List of Examples

List of Figures

List of Tables

Preface

Oracle Identity Federation is a self-contained federation server that enables single sign-on and authentication in a multiple-domain identity network.

Audience

This document is intended for administrators of Oracle Identity Federation, who will deploy and manage the operation of the server in a federated network environment.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Fusion Middleware and Oracle Database documentation sets:

- Online help available through Oracle Enterprise Manager Fusion Middleware Control
- The Oracle Fusion Middleware and Oracle Database documentation sets, especially:
  - *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*
  - *Oracle Fusion Middleware Getting Started with Oracle Identity Management*
  - *Oracle Fusion Middleware Administrator's Guide*
  - *Oracle Fusion Middleware Application Security Guide*
  - *Oracle Fusion Middleware Administrator's Guide for Oracle Directory Integration Platform*
  - *Oracle Fusion Middleware High Availability Guide*

– *Oracle Database Administrator's Guide*
– *Oracle Real Application Clusters Administration and Deployment Guide*

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.

What's New in This Guide?

This chapter provides a brief description of new features introduced with the latest release of Oracle Identity Federation, and points you to more information about each one. It contains these topics:

Note: The Oracle Identity Federation server is referred to as the federation server in this chapter.

Updates in 11g Release 1 (11.1.1.9)

This document contains the following updates in 11*g* Release 1 (11.1.1.9):

- One-to-one mapping of attribute names to assertion attributes. See Section 5.9.1.1.
- Considerations for HA configuration in SSL mode. See Section 6.4.3.
- Details and examples of post-processing plug-in configuration for Oracle Access Manager 11*g*. See Section 12.2.2, Section 12.5.3.3, and Section 12.5.3.4.
- New command syntax for entering the WLST environment. See Section 9.1.2.
- Revised command syntax for certain commands. See Section 6.20, Section 9.2.36.2, and Section 9.2.37.2.

New and Changed Features in 11g Release 1 (11.1.1.7)

This document describes the following updates in 11*g* Release 1 (11.1.1.7):

- Requirement to run the upgrade script when operating in an upgraded 11.1.1.7.0 environment for integration with Oracle Access Manager 11*g*. See Section 5.16.2.
- Support for the Backend Attribute Exchange (BAE) Direct Attribute Exchange profile. See Section 6.8.2.
- Procedure for adding an external OpenID provider. See Section 5.4.5.

New and Changed Features in 11*g* Release 1 (11.1.1.6)

This document describes the following new features and updates in 11*g* Release 1 (11.1.1.6):

- SHA-256 support for XML signatures
- Use of the Relay State in IdP-initiated SSO
- Support for Oracle Access Manager 11g authentication engine
- Support for Oracle Access Manager 11g SP integration engine
- Support for ACS URL for IdP-initiated SSO operations
- Sending ACS URL with the Authentication Request in SAML 2.0
- Implementation of OpenID UI Extension

The following additional changes appear in the document:

- Section 11.3 contains updated instructions for packaging the business processing plug-in.
- Section 6.24 explains how to configure Oracle Identity Federation to integrate with OpenID partners.
- Section 4.2.5 explains how Oracle Identity Federation determines the relay state for IdP-initiated single sign-on.
- Figure 2-11 has been corrected to show a Web proxy in front of the Oracle Identity Federation server.
- Various documentation errata have been corrected.

Updates in 11*g* Release 1 (11.1.1) Patch Set 4

This document has been updated to correct a number of documentation errata.

New Features in Oracle Identity Federation 11*g* Release 1 (11.1.1) Patch Set 3

11*g* Release 1 (11.1.1) Patch Set 3 provides these features:

- Separate keystore and encryption key passwords
- Global logout enhancements
- Configuring Oracle Identity Federation/SP to authenticate itself to Oracle Access Manager.
- Support for the OpenID protocol
- New HTTP header authentication engine
- Certification with Oracle Access Manager 11*g*
- Import and export of a provider's settings through WLST
- Restrictions on assertion validity
- The ability to override NameID mapping per partner
- Support for SAML 1.x Source ID
- Custom pre- and post-processing actions

- Support for the eAuth specifications
- Support for HTTP header collector

The following documentation changes appear in this release:

- The customization features are collected under a separate part, Part III, "Oracle Universal Federation Framework".
- The pdf version of the document is split into two volumes.

New Features In Oracle Identity Federation 11*g* Release 1 (11.1.1)

- **WebLogic Server Integration**: Oracle Identity Federation is now a Java component managed by the Oracle WebLogic Server.

  See Also:

  - Oracle Fusion Middleware Components
  - *Oracle Fusion Middleware Concepts*

- **Fusion Middleware Control**: You can manage Oracle Identity Federation by using a graphical user interface called Oracle Enterprise Manager Fusion Middleware Control

  See Also: Getting Started Managing Oracle Fusion Middleware

- **Integration with Common Auditing Infrastructure**: Oracle Identity Federation is now integrated with the Oracle Fusion Middleware audit framework. You can configure auditing from the command line or by using Fusion Middleware Control.

  See Also:

  - Section 7.4, "Auditing"
  - *Oracle Fusion Middleware Application Security Guide*

- **Support for Oracle Single Sign-On and Oracle Access Manager** 10*g* (10.1.4.2.0) or later: Oracle Fusion Middleware 11*g* Release 1 (11.1.1) does not include Oracle Single Sign-On or Oracle Access Manager. Oracle Identity Federation 11*g* Release 1 (11.1.1), however, is compatible with Oracle Single Sign-On and Oracle Access Manager 10*g* (10.1.4.2.0) or later.

Part I

Introduction

This part introduces federation concepts and introduces Oracle Identity Federation features.

Part I contains the following chapters:

1 Introduction to Oracle Identity Federation

This chapter provides an introduction to federated identity management and describes key features and benefits of Oracle Identity Federation. It contains the following sections:

- Federated Identity Management
- About Oracle Identity Federation

1.1 Federated Identity Management

Although single sign-on (SSO) enjoys wide adoption for its ability to reduce the need for redundant logins, mere SSO is insufficient for companies which must operate in a *federated* environment - that is, an environment where services must be shared with business partners while protecting those same services from unauthorized access.

A federated environment enables business partners to achieve integration in the identity management realm, by providing a mechanism for companies to share identity information across their respective security domains.

This section provides an introduction to federated identity management. It contains these sections:

- Challenges of Identity Federation
- Federation Use Cases
- Concepts
- Federation Protocols

1.1.1 Challenges of Identity Federation

Single sign-on for Web-based applications is a business goal that has been approached and solved in various ways over the past several years. Still, enterprises continue to face major challenges in managing their information systems cost-effectively. Some of these challenges include:

- The proprietary nature of many of these solutions means that the protocol and software are particular to a specific vendor, implementor, or deployment scenario, and do not readily lend themselves to easy interoperability with other single sign-on systems.
- The proliferation of content formats, supply chains, customer management systems, and user data stores poses security and maintenance concerns. For example, a financial services company serving health care organizations may need to manage hundreds of thousands of employee accounts and incur substantial

costs related to provisioning new users, responding to events like forgotten passwords, and other record maintenance. On the part of the user, disparate authentication systems mean having to remember and perhaps write down multiple IDs and passwords, with the obvious risks inherent in that practice.

- Ever-expanding and increasingly dynamic end-user communities demand that information and applications be accessible not only to employees, but to vendors, partners, and customers as well. Traditional efforts to provide access require maintaining individual user accounts within the organization, leading to duplication of identity data along with administrative and compliance issues.

Federated identity management is the evolution of the SSO paradigm in response to users' growing needs for access to computing resources and services that reside outside their own company's boundaries. In a federated environment, enterprises offering such a service can reliably obtain identity information about an individual or other entity from the user's home organization or security domain. This provides twin benefits:

1. The end user does not need to supply login credentials to access each entity where business is conducted. This also eliminates the need to remember and manage multiple logins/passwords. (Users still need accounts at the sites so that the accounts can be linked.)
2. Enterprises do not need to create additional accounts to manage the identities of users who are already known to a partner organization. In the example cited earlier, the service provider could simply leverage the employee data maintained internally by its client health care organizations.

See Also: For a detailed definition, see **federated identity management (FIM)**.

1.1.2 Federation Use Cases

Use cases in this section explain how federation can provide a seamless end-user experience by authenticating once for multiple applications, to overcome the real-world business problems of the kind described above.

Use Case 1: Single Sign-On to Partner Site

Figure 1–1 Single Sign-On from Employee Portal to Partner

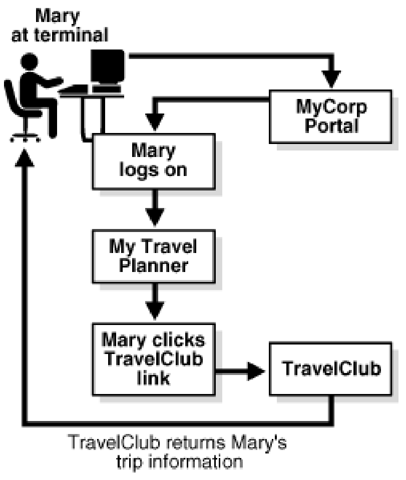

Mary at terminal
MyCorp Portal
Mary logs on
My Travel Planner
Mary clicks TravelClub link
TravelClub
TravelClub returns Mary's trip information

Figure 1–1 describes a situation where Mary, an employee of MyCorp, wishes to plan an upcoming business trip. She is able to achieve this seamlessly, in a single session, by performing the following steps:

1. Mary accesses her company's MyCorp employee portal from her terminal.
2. The portal, which is enabled with WS-Federation, presents her with a sign-on dialog.
3. After Mary signs on, the portal returns a page personalized with her information.
4. Mary commences travel planning by clicking on a link inside the portal for TravelClub, which is a partner organization providing access to a range of travel services for MyCorp employees. Mary has already established a federated relationship with TravelClub.
5. TravelClub requires authentication before Mary can access her account, and requests the same from MyCorp, which returns the necessary identity information to the travel site. Mary is then automatically authenticated to the TravelClub site. TravelClub returns a page with Mary's travel account information.
6. When Mary is done, she can log out of both her TravelClub and MyCorp sessions using a single global logout feature at the MyCorp home page.

In this way, Mary can authenticate once to her company's Web site, connect with another site and perform necessary tasks, without the need for any additional authentication at the second site.

Use Case 2: New Federated Account at Partner Site

Figure 1–2 Creating a Federated Account

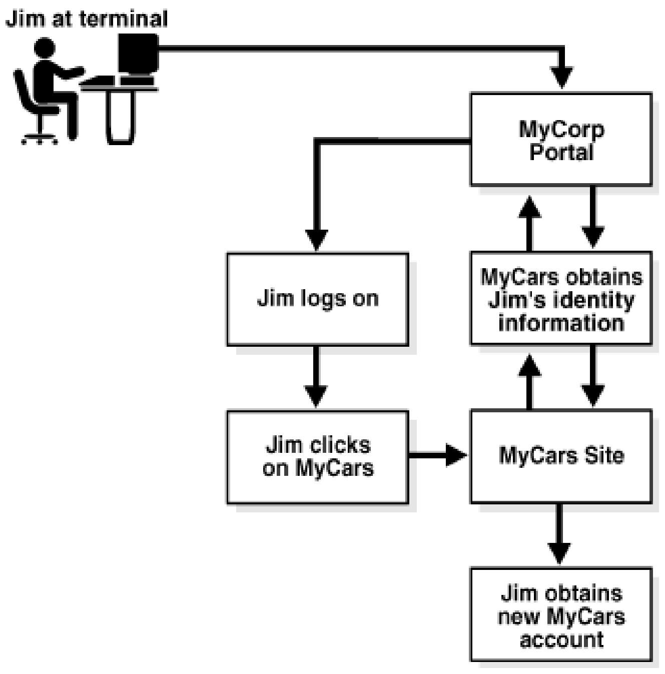

Jim at terminal
MyCorp Portal
Jim logs on
MyCars obtains Jim's identity information
Jim clicks on MyCars
MyCars Site
Jim obtains new MyCars account

Figure 1–2 illustrates a use case where Jim, another employee at MyCorp, wishes to set up a new account at MyCars, an external site which provides discount auto repair services to MyCorp employees. The steps are as follows:

1. Jim signs on to the MyCorp portal.
2. After doing some work within the portal, Jim elects to move to the "Vendors" page of the portal to look for automotive services, and clicks on the MyCars link.
3. Information is required to set up a new account at MyCars. With Jim's permission, MyCars communicates with MyCorp to obtain information relevant to Jim's identity.
4. Jim now has an account at MyCars, which he can access in a manner similar to that outlined in the previous use case.

These use cases are typical examples of the application of federated single sign-on and federated identity management. In subsequent sections we take a closer look at the key concepts of federation technology, and how they are leveraged in Oracle Identity Federation.

1.1.3 Concepts

The following discussion introduces key concepts of federated identity management.

Principal

A principal is any entity capable of using a service and capable of acquiring a federated identity.

A Principal is a person (a "user"), a group of users such as a corporation, or a system entity whose identity can be authenticated.

This is one of the three primary roles defined in the identity federation protocols supported by Oracle Identity Federation. The others are identity provider (IdP) and service provider (SP).

Domains

A **domain** is a Web site and applications that enable a principal to utilize resources. A federated site acts as an **identity provider** (source domain under some specifications), a **service provider** (destination domain), or both.

Identity Provider

An identity provider (IdP) is responsible for managing, authenticating, and asserting a set of identities within its set of federations.

This is one of the three primary roles defined in the identity federation protocols supported by Oracle Identity Federation. The others are **principal** and **service provider**.

An identity provider is sometimes also referred to as a source domain in the SAML 1.x protocols. From this perspective, it is the point at which requests originate; users from a source domain request permission to access resources on sites residing on destination domains.

Service Provider

A service provider (SP) provides services to a Principal while relying on an identity provider to authenticate the Principal's identity.

A service provider is also referred to as a relying party (SAML) or a destination domain. From the domain perspective, a service provider contains the resource that users from source domains wish to access.

Service providers are organizations offering Web-based services to users. This broad category includes practically any organization on the Web today, for example:

- internet portals
- retailers
- financial institutions
- government agencies
- not-for-profit organizations
- entertainment companies
- transportation providers

This is one of the three primary roles defined in the identity federation protocols supported by Oracle Identity Federation. The others are **identity provider** and **principal**.

Note: A single organization may be both an identity provider and a service provider, either generally or in the context of a given interaction.

OpenID Provider

A provider that performs user authentication through the OpenID protocol; abbreviated as OP. Similar to a SAML Identity Provider.

Relying Party

An OpenID Relying Party; abbreviated as RP. Similar to a SAML Service Provider.

Claimed Identifier

An OpenID identifier that refers to a user managed by the OP.

Association

An agreement between an OP and RP that defines a secret MAC key to be used for message signatures.

Discovery

An online means by which the OpenID OP and RP can determine and verify service support and endpoint locations. Discovery in OpenID is done through the XRDS metadata publishing protocol. An OP's XRDS metadata advertises its supported services (for example, attribute exchange, PAPE, and so on.) and its sign-on URL. An RP's XRDS advertises its return URL.

Federations

A federation is a trust relationship between an identity provider and a service provider, which allows a principal to use a single federated identity and single sign-on when conducting business transactions with those providers.

Organizations create federations based on federation technology and operational agreements that define trust relationships between them.

For example, an enterprise federation could be created where the identity provider is a company leveraging employee network identities across the enterprise. Another example is in consumer banking, where the user's bank has established business relationships (that is, federations) with various other service providers, allowing the user to wield his bank-based network identity with those providers.

Federation Solutions

The essence of federation is the exchange of identity data between independent security domains. When we think about how to achieve federation, we inevitably must think about trust levels, goals and objectives, and the amount of administration involved in different types of federation:

> **Note:** This list defines ways of thinking about federation at a business level, and about the types of solutions that may be relevant in planning your federation; they do not necessarily correspond to specific technical solutions.

- Transient Federation

  Transient federation involves the transfer of the user session across domains without the need to send or consume additional identity data. The receiver relies on the sender to authenticate/authorize the principal.

  Transient federation is suited for situations where a) the providers implicitly trust each other, and b) they agree to support common standards such as SAML 2.0.

  Transient federation is relatively easy to implement and administer; the user experience is transparent as the user simply clicks a link to be able to access the service provider as an authenticated user.

  In a common application of transient federation, a government agency - which maintains and authenticates citizen information in its own domain using certificates and other authentication methods - federates the user to other agencies to enable single sign-on to their services.

  An obvious limitation of transient federation is the inability of the receiving domain to perform additional user verification due to reliance on the sending domain.

- Account Mapping

  Account mapping is suited for situations where the security domains may not have a very high level of established trust, or they wish to limit the information accessible to the federated user.

  Account mapping involves additional administration since identities must be maintained in both domains. The domains exchange user information (assertions) in the form of messages that contain a user ID that can be mapped by the receiver to a known identity in its local store. By the same token, account mapping offers the advantage of not requiring a high level of trust between federating domains.

- Account Linking

  Account linking is an extension of account mapping; however, instead of maintaining local identity details in each domain, a receiving partner can utilize, or extend, the information it obtains from a user the first time federation occurs, and store it for future use.

  The task of account creation is relegated to the user, and thus account linking reduces the administration effort needed by account mapping.

  A typical application of account linking and account mapping is seen in the travel industry. Users can plan all aspects of their trip, including airline reservations, car rental, and hotel, when the businesses in question federate with related partners to share business opportunities, and also enable the user to access the necessary services with a minimum of authentications.

- Attribute Federation

  Attribute federation is useful in situations where the partners want or need to grant access to their resources based on access control policies.

  Instead of maintaining user identities, each domain maintains such information as groups, roles, and so on. During federation, a receiving partner examines the user's attributes and maps them to this authorization information to determine what type of access to allow. Thus, the partners must agree on a common set of mapping rules.

  Administration is simpler (than with account mapping/linking) since only rules must be maintained, not actual identity data.

  An example of this type of federation is a business-to-business (B2B) environment, where the partners exchange identity information while maintaining their respective rules to grant access and determine the authorization for users requesting access to their resources.

- Combined Federation

  Combined federation, as the name suggests, enables the sending domain to provide as much identifying information (name, address, role, and so on) as it wishes in the federation request.

  An advantage of this federation approach is the flexibility it provides each domain in how it formulates the request. At the same time, a disadvantage of combined federation is the high administration needs and the requirement of strong trust relationships and coordination among partners.

Identity Federation

Identity federation is the linking of two or more accounts a principal may hold with one or more IdPs or service providers that have created a federation.

When users federate the otherwise isolated accounts they have with businesses (known as their local identities), they are creating a relationship between two entities, an association comprising any number of service providers and identity providers.

From a technical perspective, a principal's identity is said to be federated between a set of providers when there is an agreement between the providers on a set of identifiers and/or attributes to use to refer to the principal.

Single Sign-On

Single sign-on enables users to sign on once with a member of a federated group of identity providers and service providers and subsequently use various resources among the group without the need to sign on again.

Under the SAML or WS-Federation protocols, performing a single sign-on operation between a principal, an SP and an IdP requires that:

- there exists a federation between the SP and IdP, that is, they have a trusted business relationship.
- the principal has local identities (or roles) as both the SP and the IdP, and that the two identities are federated.

Oracle Universal Federation Framework

The Oracle Universal Federation Framework provides out-of-the-box support for industry-standard federation features. It also provides a number of extensibility features, including:

- custom actions before and after authentication and SP integration processing (this feature is also referred to as pre- and post-processing plugins)
- custom authentication engines
- custom SP integration engines

1.1.4 Federation Protocols

In building a federated architecture that addresses interoperability, assurance, and trust concerns across security domains, the following protocols have emerged as useful building blocks for identity management integration:

- SAML 1.0 and 1.1, which define a format for security data exchange known as an assertion, and profiles which provide the means for using the assertions
- SAML 2.0, which extends SAML 1.1 to provide additional profiles.
- WS-Federation, which enables different security realms to federate by brokering trust of identities, user attributes, authentication between participating Web services

The SAML and WS-Federation protocols provide a framework for exchanging information between security domains, for provider introduction or "handshake," and for managing identity events such as federated sign-on and global logout. These standards provide an XML-based framework for communicating security information across domains. Benefits include:

- loose coupling of domains, with no need to synchronize or replicate user data between directories
- a platform- and technology-neutral approach that removes barriers to cross-domain integration
- broad support from application server, web services management, and security products and vendors, and adoption by a growing number of organizations

This section explains fundamental SAML concepts and provides details about federation protocols. It contains these topics:

- SAML Basics
- Evolution of the Federated Identity Standards
- SAML 1.x
- SAML 2.0
- WS-Federation

See Section 1.2.4, "Federation Protocol Profiles" for a description of the profiles supported in Oracle Identity Federation.

Note: Liberty 1.x support is deprecated.

1.1.4.1 SAML Basics

Security Assertions Markup Language (SAML), a standard developed by OASIS, provides a means for exchanging security information between online business partners.

In a typical exchange of SAML messages between two parties, one party acts as a relying party while the other acts as an asserting party.

The asserting party *asserts* information about a given subject, such as whether a subject has been authenticated, whether a subject is authorized to perform a certain action, and so on. The relying party uses information provided by the asserting party to make security-related decisions regarding a subject, such as what types of access to grant the subject to a specific resource, and so on.

SAML Assertions

SAML associates a principal with additional identity information that can be used to determine the principal's access rights within a specific domain. Every SAML document has an `assertion` element containing such an association.

SAML defines three kinds of assertions, which are declarations of one or more facts about a subject:

- authentication assertions, which state that the user has proven her identity by a particular method at a specific time
- attribute assertions, which contain specific details about the user such as an employee number or an account number
- authorization assertions, which state the resources a user can access and under what conditions they can be accessed

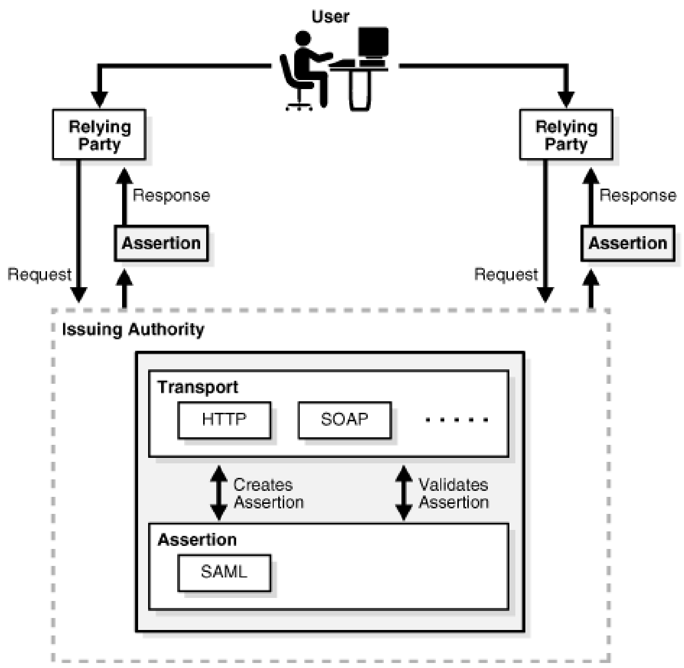
Assertions are coded statements generated about events that have already occurred. While SAML makes assertions about credentials, it does not actually authenticate or authorize users.

Example 1–1 shows a typical SAML 1.0 authentication assertion wrapped in a SAMLP response message:

SAML Request and Response Cycle

In a typical SAML cycle, the relying party, which needs to authenticate a specific client request, sends a SAML request to its issuing authority. The issuing authority responds with a SAML assertion, which supplies the relying party with the requested security information. This cycle is illustrated in Figure 1–3.

Figure 1–3 The SAML Request-Response Cycle



For example, when a user signs into a SAML-compliant service of a relying party, the service sends a "request for authentication assertion" to the issuing authority. The issuing authority returns an "authentication assertion" reference stating that the user was authenticated by a particular method at a specific time. See Example 1–1 for a typical authentication assertion.

The service can then pass this assertion reference to other relying party sites to validate the user's credentials. When the user accesses another SAML-compliant site that requires authentication, that site uses the reference to request the "authentication assertion" from the issuing authority, which states that the user has already been authenticated.

At the issuing authority, an assertion layer handles request and response messages using the SAML protocol, which can bind to various communication and transport protocols (HTTP, SOAP, and so on). Note that while the client always consumes assertions, the issuing authority can act as producer and consumer since it can both create and validate assertions.

SAML Protocol Bindings and Profiles

SAML defines a protocol for requesting and obtaining assertions (SAMLP). Bindings define the standard way that SAML request and response messages are transported across the issuing authorities and relying parties by providing mappings between SAML messages and standard communication protocols. For example, one defined transport mechanism for SAML requests and responses over HTTP is Simple Object Access Protocol (SOAP). This enables the exchange of SAML information across several Web services in a standard manner.

A profile describes how SAML assertions are embedded into and extracted out of standard frameworks and protocols. Web browser profiles for single sign-on and SOAP profiles for securing SOAP payloads are some of the available profiles.

1.1.4.2 Evolution of the Federated Identity Standards

In response to the needs of access control vendors for a standard mechanism to speed development of cross-domain single sign-on applications, efforts by the OASIS standards organization produced SAML 1.0, the first such standard, in 2002. The Liberty Alliance, working on open standards for federated identity, built upon the SAML specifications to produce the Liberty 1 standard. At the same time, another consortium of vendors and companies worked on evolving authentication and authorization standards for Web service-oriented applications. Subsequent efforts led to the development of the WS-Federation standard, and extension and co-evolution of the SAML and Liberty standards in parallel. These different standards are described below.

1.1.4.3 SAML 1.x

SAML 1.0 defines two key concepts:

1. a security token format, known as an assertion, which associates a given identity with specific access rights
2. profiles that describe ways to package these assertions to provide single sign-on

SAML 1.1 updates SAML 1.0 with feedback and corrections. Specifically, SAML 1.1 introduces XML Digital Signatures changes that greatly improve interoperability. Because of these XML Digital Signature changes, Oracle recommends that you use the SAML 1.1 protocol over SAML 1.0 whenever possible as it greatly reduces issues when verifying signatures.

Example 1–1 shows a typical SAML 1.0 authentication assertion wrapped in a SAMLP response message:

Example 1–1 Sample SAMLP Response Containing a SAML 1.0 Authentication Assertion

```
<samlp:Response
       MajorVersion="1" MinorVersion="0"
       ResponseID="128.14.234.20.90123456"
       InResponseTo="123.45.678.90.12345678"
       IssueInstant="2005-12-14T10:00:23Z"
       xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
       xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol">
       <samlp:Status>
           <samlp:StatusCode Value="samlp:Success" />
       </samlp:Status>
       <saml:Assertion
           MajorVersion="1" MinorVersion="0"
           AssertionID="123.45.678.90.12345678"
           Issuer="IssuingAuthority.com"
           IssueInstant="2005-12-14T10:00:23Z" >
           <saml:Conditions
               NotBefore="2005-12-14T10:00:30Z"
               NotAfter="2005-12-14T10:15:00Z" />
           </saml:Conditions
           <saml:AuthenticationStatement
               AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
               AuthenticationInstant="2005-12-14T10:00:20Z">
               <saml:Subject>
                   <saml:NameIdentifier NameQualifier="RelyingParty.com">
                     john.smith
                   </saml:NameIdentifier>
                   <saml:SubjectConfirmation>
                       <saml:ConfirmationMethod>
```

```
                           urn:oasis:names:tc:SAML:1.0:cm:artifact-01
                       </saml:ConfirmationMethod>
                   </saml:SubjectConfirmation>
               </saml:Subject>
           </saml:AuthenticationStatement>
       </saml:Assertion>
</samlp:Response>
```

1.1.4.4 SAML 2.0

SAML 2.0 includes support for single sign-on based largely on the framework developed by the Liberty Alliance ID-FF specifications.

Although the concept of identity federation is not present in the specifications, SAML 2.0 promotes the existence of a name identifier for a specific use. SAML 2.0 supports a number of named profiles that largely mirror the functionality of the Liberty ID-FF 1.2 profiles, on top of the name identifiers inherited from SAML 1.x.

Example 1–2 shows a SAML 2.0 authentication assertion:

Example 1–2 SAML 2.0 Authentication Assertion

```
<saml:Assertion
       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
       Version="2.0"
       ID="id-V01vmFAGUOKmKVJh9-hQ-gsPhX8-"
       IssueInstant="2005-10-06T21:03:17.375Z">
       <saml:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
           http://issuingauthority.example.com/
        </saml:Issuer>
        <!-- signature by the issuer over the assertion -->
       <ds:Signature>
           ...
       </ds:Signature>
       <saml:Subject>
           <saml:NameID
             Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
               id-V9l9N2S4nA8KlHd0X9Df3KYKm4E-
           </saml:NameID>
           <saml:SubjectConfirmation Method=
             "urn:oasis:names:tc:SAML:2.0:cm:bearer">
               <saml:SubjectConfirmationData
                 NotOnOrAfter="2005-10-06T21:03:32.375Z"
                 Recipient="http://issuingauthority.example.com/fed/sp/art20"
                 InResponseTo="id-G2mgYgtGH9gu8Nwo8KwxPYrpXKE-"/>
             </saml:SubjectConfirmation>
       </saml:Subject>
       <saml:Conditions NotBefore="2006-04-27T16:40:49Z"
         NotOnOrAfter="2006-04-27T17:05:49Z">
           <saml:AudienceRestriction>
               <saml:Audience>
                 http://serviceprovider.example.com:80/fed/sp
               </saml:Audience>
           </saml:AudienceRestriction>
       </saml:Conditions>
       <saml:AuthnStatement
           AuthnInstant="2005-10-06T21:01:03.451Z"
           SessionIndex="1448745">
           <saml:AuthnContext>
               <saml:AuthnContextClassRef>
```

```
                    urn:oasis:names:tc:SAML:2.0:ac:classes:
                      PasswordProtectedTransport
                </saml:AuthnContextClassRef>
            </saml:AuthnContext>
        </saml:AuthnStatement>
</saml:Assertion>
```

1.1.4.5 WS-Federation

The WS-Federation specification is "an integrated model for federating identity, authentication, and authorization across different trust realms and protocols." WS-Federation is a Web services-oriented standard which supports profiles for passive requesters, such as Web browsers, and active requesters such as SOAP-enabled applications.

Note: Oracle Identity Federation currently supports passive requesters for WS-Federation.

1.2 About Oracle Identity Federation

This section shows how Oracle Identity Federation allows users of Oracle Identity Management products, and customers new to the Oracle APS stack, to engage in business associations across heterogeneous environments using various sources of user authentication. It contains the following topics:

- Features and Benefits of Oracle Identity Federation
- Architecture
- High-Level Processing Flow
- Federation Protocol Profiles
- Affiliations
- Cryptographic Provider
- Example of Federation Event Flow
- Supported Standards and Applications

1.2.1 Features and Benefits of Oracle Identity Federation

Oracle Identity Federation is a standalone, self-contained federation server that enables single sign-on and authentication in a multiple-domain identity network. Oracle Identity Federation supports multiple federated identity protocols including the Liberty ID-FF, OpenID, and SAML protocols. This allows users to federate in heterogeneous environments and business associations, whether or not they have implemented other Oracle Identity Management products in their solution set.

Key features of Oracle Identity Federation include:

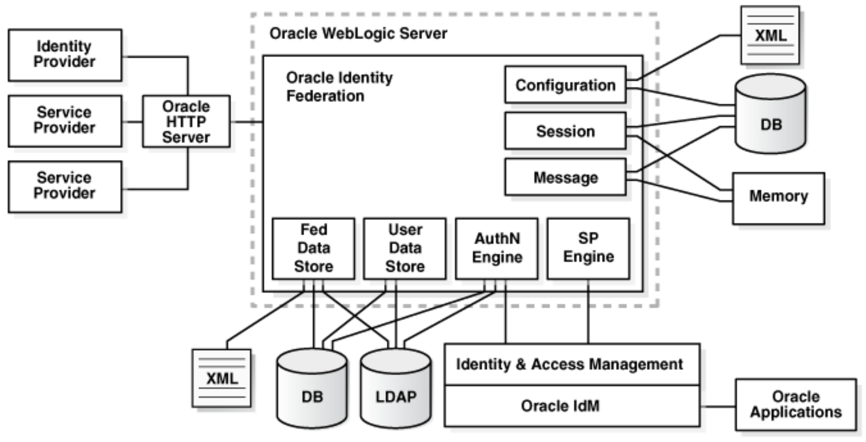
- the ability to implement cross-site access and authentication in an environment containing both identity providers and service providers
- the ability to configure, enable, and disable external sites
- the ability to access applications at destination sites using a single sign-on
- support for these leading federation protocols:

  - SAML 1.x, including SAML 1.x attribute requester and responder, authn query responder and assertion ID responder functionality
  - SAML 2.0, including SAML 2.0 attribute requester and responder, authn query responder and assertion ID responder functionality
  - WS-Federation passive requester
  - Liberty ID-FF 1.x
  - OpenID 2.0
- integration with Oracle Access Manager and Oracle Single Sign-On
- support for cross-protocol single sign-on and sign-out
- support for affiliations, which reduce the number of federations by allowing service providers to share their federation information
- integration with Oracle Internet Directory and support for:
  - a range of authentication engines, including Oracle Access Manager and LDAP
  - user data repositories, including LDAP Stores such as Microsoft Active Directory and Sun Java System Directory Server
  - relational databases
- support for X.509 certificate validation

1.2.2 Architecture

Figure 1–4 shows the architecture of Oracle Identity Federation and its relationship to other federation components. Here Oracle Identity Federation (denoted as OIF) has created federations with other identity providers and service providers, which can be additional Oracle Identity Federation instances or third-party providers.

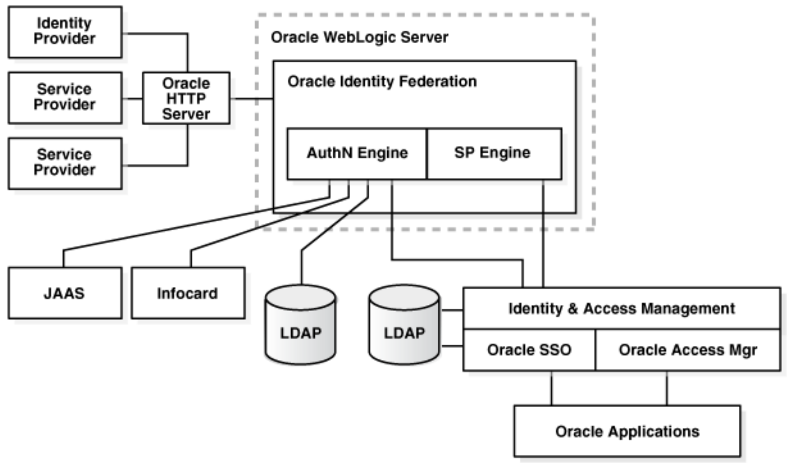
Figure 1–4 Oracle Identity Federation



Note: A single Oracle Identity Federation instance can act as an IdP and an SP. The identity provider and the two service providers shown in the figure are federated peer providers.

Oracle Identity Federation includes a self-contained, lightweight authentication service.

Figure 1–5 Oracle Identity Federation 3rd Party Integration



Oracle Identity Federation can communicate with a range of authentication mechanisms and user data repositories:

1. Oracle Identity Management

   You can configure the Oracle Identity Federation authentication service to enable single sign-on access to resources protected by Oracle Single Sign-On or Oracle Access Manager, including:

   - Oracle Collaboration Suite
   - Oracle E-Business Suite
   - PeopleSoft modules
   - and more

   In addition to Oracle Single Sign-On (with the Oracle Internet Directory user repository) or Oracle Access Manager (with various repositories), this configuration can also leverage third-party access management solutions when Oracle Single Sign-On is deployed for use with those solutions.

   Note: In an environment where Oracle Identity Federation and Oracle Single Sign-On both protect resources, you can configure either component to serve as the authentication mechanism when a user requests access to a protected resource. For example, Oracle Identity Federation can forward authentication requests to Oracle Single Sign-On; or, Oracle Single Sign-On can request Oracle Identity Federation to locate an appropriate identity provider.

   Likewise, environments containing both Oracle Identity Federation and Oracle Access Manager provide similar functionality.

2. Data Stores

   You can configure Oracle Identity Federation to access:

   - LDAP directories

- RDBMS databases

1.2.3 High-Level Processing Flow

Before looking at Oracle Identity Federation features in detail, it is instructive to consider, at a high level, the processing flow that makes it possible to manage user access in such a federated environment.

Users typically access applications in multiple domains through a corporate portal. For example, Alpha Corporation could have a Portal Server in place to manage Alpha's user logins, page personalization, and so on. The portal server might consist of homegrown logic running within an application server, or it might be a commercial product. Its partner, Beta Corporation, may serve its technical database application with a "MyBeta.com" type of portal. In that case, each company would operate its own portal server.

The processing flow is as follows:

1. The user logs into Alpha portal whose access is being managed by a web access management (WAM) system like Oracle Access Manager, Oracle Single Sign-On or other product.
2. The user initiates a request, by clicking on a resource that is actually hosted by Beta corporation.
3. The Oracle Identity Federation instance at Alpha portal acting as the identity provider (IdP) sends the user information to the WAM system.
4. The WAM system creates a session after mapping a user to an identity in its local identity store.
5. The WAM system returns a successful response and the session token to the Oracle Identity Federation IdP server at Alpha portal.
6. Using the above information, the IdP at Alpha portal creates a SAML identity assertion and signs it using its private signing key. This response is sent to the Oracle Identity Federation instance acting as a Service Provider (SP) at Beta Corporation.
7. The Oracle Identity Federation server acting as SP at Beta Corporation verifies the signed response using the IdP's public certificate associated with its signing key.
8. The Oracle Identity Federation service provider at Beta Corporation extracts the assertion, and creates a user session for the assertion after mapping the user session to its local authorization system.
9. The Oracle Identity Federation service provider sends the user's browser a redirect to the requested resource.
10. The user's browser sends a request to the target resource over the user session created by the service provider.

For a more detailed description of Oracle Identity Federation processing flows, see Chapter 2, "Planning Oracle Identity Federation Deployment".

1.2.4 Federation Protocol Profiles

Identity providers and service providers exchange assertions using profiles and services defined in a federation protocol such as SAML, WS-Federation, or OpenID. Assertion functions include:

- establishing secure connections

- conveying authentication data across those connections
- receiving and interpreting assertions from other domains

Profiles describe the types of exchanges required to transfer assertions between IdP and SP. This section takes a closer look at the assertion profiles available in Oracle Identity Federation:

- Browser POST Profile
- Browser Artifact Profile
- SOAP Binding
- Browser HTTP Redirect Profile
- Name Identifier Management Profiles
- SAML Attribute Sharing Profile
- WS-Federation Passive Requester Profile
- Federation Termination Profile
- Global Logout Profile
- OpenID Profiles and Extensions

Note: Liberty 1.x support is deprecated.

1.2.4.1 Browser POST Profile

The SAML Browser POST profile sends a full assertion from an identity provider to a service provider without the use of an artifact. Oracle Identity Federation sends the assertion to the user's browser as a hidden variable in the HTML form, and the browser then posts the assertion to the destination site.

In SAML and WS-Federation, the HTTP POST Binding provides a framework for the embedding and transport of SAML protocol messages under real-world communication protocols.

1.2.4.2 Browser Artifact Profile

Some browsers may limit the number of URL characters they can handle. The SAML Browser Artifact profile accommodates this by transmitting data using a compact reference to an assertion, called an artifact, instead of sending the full assertion. The recipient of the artifact then uses an artifact resolution protocol to obtain the full assertion referred to by the artifact.

In SAML, the HTTP Artifact Binding provides a framework for the embedding and transport of artifacts under real-world communication protocols.

1.2.4.3 SOAP Binding

A binding is the mapping of abstract message exchanges into real-world messaging or communication protocols. As an example, the SAML SOAP Binding defines how SAML protocol messages can be communicated within SOAP messages.

1.2.4.4 Browser HTTP Redirect Profile

The Browser HTTP Redirect profile indicates to the requesting party that the requested resource resides under a different URL.

In SAML and WS-Federation, the HTTP Redirect Binding uses the HTTP redirect response to send data in URL query string parameters through a user's browser from one provider to another. The amount of data that can be sent is limited by the maximum URL allowed by the browser, so this is usually employed for shorter messages and not full assertions.

1.2.4.5 Name Identifier Management Profiles

Name Identifier Profiles define how providers communicate with each other when one of the providers wishes to update the name identifier assigned to one of their common users. These profiles allow a service provider or identity provider to specify (or register) a *name identifier* for a principal. Peer providers, for their part, must use this name identifier when communicating with other providers about the principal.

Oracle Identity Federation supports these SOAP/HTTP and HTTP-redirect name identifier profiles:

- Liberty ID-FF 1.1 IdP-Initiated Register Name Identifier Profile
- Liberty ID-FF 1.1 SP-Initiated Register Name Identifier Profile
- Liberty ID-FF 1.2 IdP-Initiated Register Name Identifier Profile
- Liberty ID-FF 1.2 SP-Initiated Register Name Identifier Profile
- SAML 2.0 IdP-Initiated Manage NameID Profile for Name Identifier Update
- SAML 2.0 SP-Initiated Manage NameID Profile for Name Identifier Update

1.2.4.6 SAML Attribute Sharing Profile

SAML provides an Attribute Query/Response protocol for retrieving a principal's attributes.

To see how this is protocol is used, consider a principal who needs to access a web resource maintained at a service provider. Authentication is achieved by presenting the user's federated credential in the form of a trusted X.509v3 certificate, along with proof of possession of the associated private key. One common example of this is the client certificate authentication feature of the SSL (Secure Sockets Layer) protocol used between a user's browser and a web server.

The service provider may require additional information about the principal to determine authorization for some privileged resource. To get this information, the SP utilizes the `SubjectDN` from the principal's X.509v3 certificate to query an identity provider for the required attributes. When the IdP returns these attribute values, the SP can make an authorization decision based on the additional data. Thus, the profile provides additional protection of resources from unauthorized access.

1.2.4.7 WS-Federation Passive Requester Profile

WS-Federation provides support for integration of identity, authentication, and authorization across security domains and protocols. The WS-Federation passive requester profile defines the use of this specification when clients for federation services include such passive requesters as Web browsers that support the HTTP protocol.

1.2.4.8 Federation Termination Profile

Users have the ability to terminate a federation, typically by using a link on the identity provider's or service provider's Web site. If initiated at the IdP, this action tells the SP that the IdP will no longer provide the user's identity information to the SP. If initiated at the SP, this action tells the IdP that the user requests that the IdP will no longer provide that user's identity information to the SP.

Note: Federation termination is also referred to as defederation.

The Federation Termination Profile specifies how identity providers and service providers are notified of federation termination.

Oracle Identity Federation supports these federation termination profiles:

- Liberty ID-FF 1.1 IdP Initiated Federation Termination Notification Profile
- Liberty ID-FF 1.1 SP Initiated Federation Termination Notification Profile
- Liberty ID-FF 1.2 IdP Initiated Federation Termination Notification Profile
- Liberty ID-FF 1.2 SP-initiated Federation Termination Notification Profile
- SAML 2.0 IdP Initiated Manage NameID Profile for Name Identifier Deletion
- SAML 2.0 SP Initiated Manage NameID Profile for Name Identifier Deletion

1.2.4.9 Global Logout Profile

As the name implies, this profile provides support for global logout. The identity provider maintains a list of all the service providers at which a given user has logged in based on assertions provided by the IdP. When the user invokes logout, the IdP sends each SP a logout request for the user, achieving global logout with respect to that IdP.

The steps in the logout process are:

1. Either the user or a peer provider initiates the logout request.

   The Oracle Identity Federation IdP sends a logout request to the service providers or identity providers where the user was logged in. The type of message sent depends on the type of single sign-on.
2. Oracle Identity Federation receives a logout response from the provider to whom it sent the message.
3. Oracle Identity Federation sends the next logout request (step 1).
4. When the user is logged out of all the providers, Oracle Identity Federation logs the user out of the server.
5. For WS-Federation logout, Oracle Identity Federation displays a success page to the user. SAML 2.0 logout profiles send the user back to the requesting peer provider that sent the original logout request.

Oracle Identity Federation supports SOAP/HTTP and HTTP-Redirect global logout profiles for these protocols:

- SAML 2.0 IdP-Initiated Single Logout Profile
- SAML 2.0 SP-Initiated Single Logout Profile
- WS-Federation Passive Requester Logout Profile
- Liberty ID-FF 1.1 IdP-Initiated Single Logout Profile
- Liberty ID-FF 1.1 SP-Initiated Single Logout Profile
- Liberty ID-FF 1.2 IdP-Initiated Single Logout Profile

- Liberty ID-FF 1.2 SP-Initiated Single Logout Profile

1.2.4.10 OpenID Profiles and Extensions

Oracle Identity Federation supports the following OpenID extension and profile specifications:

- Attribute Exchange (AX)

  AX is an OpenID 2.0 extension allowing user attributes to be requested and returned.

- Provider Authentication Policy Extension (PAPE)

  PAPE is an OpenID 2.0 extension allowing RPs to request specific authentication type and strength, including Levels of Assurance.

- US Government Federal Identity, Credentialing and Access Management (ICAM) Profile

  This profile supports policy and security requirements for the US Government for OpenID 2.0 deployments, including:

  – No Personal Identification Information

  – Private Personal Identifier

  – GSA Profile for OpenID

  – NIST authentication levels

For details, see OpenID topics in Section 2.2, "Profiles and Bindings".

1.2.5 Affiliations

A SAML 2.0 or Liberty 1.2 affiliation consists of service providers that are part of a logical group.

Note: Liberty 1.x support is deprecated.

An affiliation is not a concrete entity or server, but a logical grouping of providers. Thus, no particular server can act as an affiliation; rather, the affiliation identity is used by service providers when performing protocol message exchanges. In this case, the messages appear to come from the affiliation logical entity, but the actual sender of the messages is a specific service provider. In this way, the affiliation serves as an alias for all service providers in the affiliation, each of which makes use of the federation information associated with the affiliation entity.

1.2.6 Cryptographic Provider

Oracle Identity Federation uses Java Cryptographic Extension for any signature and encryption operations it needs to perform.

The signing and encryption keys must be provided to Oracle Identity Federation either as a PKCS#12 wallet, or as a Java Keystore.

Note: It is possible to configure Oracle Identity Federation to use a specific JCE Provider that will provide signing and encryption keys.

1.2.7 Example of Federation Event Flow

This section describes a typical message flow in a federated interaction.

Elaborating on the use case in Figure 1–1 on page 1-3, consider that Mary is already authenticated at mycorp.com, and goes to travelclub.com where she is not logged in. travelclub.com requires Mary to be authenticated before she can access her local account, and redirects Mary with a SAML 2.0 message to mycorp.com requesting a single sign-on for travelclub.com. Since Mary is already logged in at the identity provider, mycorp.com retrieves Mary's account and federation data and redirects her back to travelclub.com. Using the Provider Identifier `mycorp.com` and the User Identifier `xyz123` provided with the redirect, travelclub.com can uniquely retrieve Mary's federation data and her local account.

1.2.8 Supported Standards and Applications

For information about the platforms and product versions supported by Oracle Identity Federation, see the appropriate certification matrix as follows:

1. Log in to My Oracle Support at `https://metalink.oracle.com`.
2. Click the **Certify** tab.
3. Click **View Certifications by Product**.
4. Select the Application Server option and click **Submit**.
5. Select the Oracle Identity Management option and click **Submit**.
6. Under General Oracle Identity Management Certification Information, click Oracle Identity Federation Certification.

2 Planning Oracle Identity Federation Deployment

This chapter outlines Oracle Identity Federation deployment considerations and helps you understand installation options. It contains these sections:

- Architecture Options
- Profiles and Bindings
- Authentication Engines
- Data Repositories
- Installation Requirements
- Sizing Guidelines
- Implementation Checklist

2.1 Architecture Options

In planning to deploy Oracle Identity Federation, you should understand the server architecture, the operating environment, and the role that your server will play in a federated exchange network. This section outlines the architectural aspects of Oracle Identity Federation deployment, including:

- Role in Federation
- Proxy Server
- Server Security
- Protocol

2.1.1 Role in Federation

As described earlier, an Oracle Identity Federation instance in a federated network can serve as an identity provider (IdP), a service provider (SP), or both.

Identity Provider Role

When a user wishes to access a protected resource in the federated network, the service provider for that resource directs the user to Oracle Identity Federation, which acts as **identity provider** for authentication. Oracle Identity Federation uses an authentication engine to obtain credentials and authenticate the user. Oracle Identity Federation can now assert the user's identity to the resource (SP), which authenticates the user and provides the requested application.

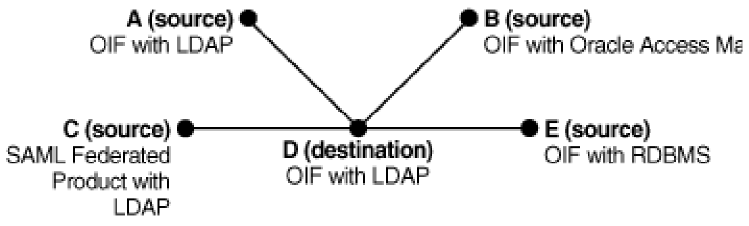
Service Provider Role

A user tries to access a resource protected by an authentication engine such as Oracle Single Sign-On, which redirects the user to Oracle Identity Federation. In a **service provider** role, Oracle Identity Federation redirects the user to an identity provider such as a portal for global authentication. The IdP portal can now obtain credentials, authenticate the user, and redirect back to Oracle Identity Federation, which then retrieves the asserted identity from the IdP. Oracle Identity Federation redirects the (authenticated) user to the authentication engine, which grants access to the protected resource.

Federation Topology

A federation can comprise any number of identity providers and service providers. One common federation topology is referred to as the hub-and-spoke model. In this topology, there is either a single service provider accepting authentication from multiple identity providers, or a single identity provider authenticating to multiple service providers.

Figure 2–1 A Hub-and-Spoke Federation Network



2.1.2 Proxy Server

You must decide what components you will put in the DMZ and whether to use a proxy server. If you put Oracle Identity Federation behind the fire wall, the proxy must forward requests and responses to the federation server, enabling transparent access to the server from an external network such as the internet.

Oracle Identity Federation configuration varies depending on the type of profile being implemented.

> **See Also:** For more information about setting up a proxy server for Oracle Identity Federation, see Appendix B, "Using Oracle HTTP Server as a Proxy for Oracle Identity Federation".

POST Profile with Proxy in SP DMZ

The POST profile sends the full assertion to the SP over HTTPS. Both IdP and SP are configured to communicate through their SSL ports. When using the POST profile in production, the SP uses a proxy server in the DMZ.

Artifact Profile with Proxy in IdP and SP DMZ

When using the browser artifact profile, the IdP sends an artifact (an identifier) rather than an actual assertion. The SP receives the artifact and requests the full assertion thereafter.

If you elect to use a proxy, note that proxies must be used for both IdP and SP in order to implement this profile. The proxies serve as receiver and responder services, handling the exchange of artifacts, assertion requests and assertions, and forwarding those objects to their respective providers.

2.1.3 Server Security

Oracle Identity Federation provides secure communication using:

- SSL Encryption
- Certificate-based Authentication
- Certificate Repository and Validation

2.1.3.1 SSL Encryption

Oracle Identity Federation provides secure SSL communication between partner domains. SSL encryption is an option you can enable or disable for the server instance at installation time.

Note: For more information about SSL configuration, see Section 8.1.1.1, "Setting up SSL on Oracle WebLogic Server"

2.1.3.2 Certificate-based Authentication

For initial setup and testing, identity providers and service providers can use default self-signed certificates. Before going into production, however, you will want to ensure that your installation is set up to use third-party CA certificates.

2.1.3.3 Certificate Repository and Validation

Oracle Identity Federation provides a repository where you can store a list of trusted CAs and certificate revocation lists (CRLs).

If certificate validation is enabled for the server, Oracle Identity Federation will validate every certificate used to verify incoming signatures for the SAML and WS-Federation protocols.

To validate a certificate, the server tries to locate the certificate or its issuer as a trusted certificate, and checks that the certificate is not in a CRL.

See Also:

- Section 5.10.2, "Security and Trust - Provider Metadata" for information about enabling certificate validation
- Section 5.10.1, "Security and Trust - Wallet" for details about the certificate repository

2.1.4 Protocol

When installing Oracle Identity Federation, you must decide the federation protocols that your server will support. Oracle Identity Federation works with these protocols:

- SAML 1.0
- SAML 1.1
- SAML 2.0
- WS-Federation
- OpenID

As the Oracle Identity Federation administrator, you must determine which federation protocols you will utilize for your server.

For more information, refer to Section 1.1.4, "Federation Protocols".

2.2 Profiles and Bindings

This section discusses profiles and bindings, and contains these topics:

- Supported Protocols
- Choosing a Profile

2.2.1 Supported Protocols

Having selected the protocol(s) your federation server instance will support, you must choose which protocol profiles, security transport bindings, and other features you will implement. This section outlines Oracle Identity Federation protocol support.

2.2.1.1 SAML 2.0 Protocol

Table 2–1 shows the SAML 2.0 protocol profiles and security transport binding combinations that Oracle Identity Federation supports.

Table 2–1 Oracle Identity Federation Profiles and Bindings for SAML 2.0

Function	Profiles/ Bindings	SAML 2.0
Single Sign-On	Artifact	x
Single Sign-On	HTTP Post	x
Logout	HTTP Redirect	x
Logout	HTTP Post	x
Name ID Registration	HTTP Redirect	x
Name ID Registration	HTTP Post	x
Name ID Registration	SOAP	x
Federation Termination	HTTP Redirect	x
Federation Termination	HTTP Post	x
Federation Termination	SOAP	x
Attribute Retrieval	SOAP	x

2.2.1.2 SAML 1.x and WS-Federation Protocol

Table 2–2 shows the SAML 1.x and WS-Federation (WS-Fed) protocol profiles and security transport binding combinations that Oracle Identity Federation supports.

Table 2–2 Oracle Identity Federation Profiles and Bindings for SAML 1.x and WS-Federation

Function	Profiles/ Bindings	SAML 1.0/1.1	WS-Federation
Single Sign-On	Artifact	x	
Single Sign-On	HTTP Post	x	x

Table 2–2 (Cont.) Oracle Identity Federation Profiles and Bindings for SAML 1.x and WS-Federation

Function	Profiles/ Bindings	SAML 1.0/1.1	WS-Federation
Logout	HTTP Redirect		x

2.2.1.3 OpenID 2.0 Protocol

Oracle Identity Federation supports OpenID version 2.0.

Authentication

Oracle Identity Federation supports basic OpenID authentication request functionality.

Associations

Oracle Identity Federation supports provider federations using an HMAC shared secret for message signing and verification, including:

- HMAC-SHA1
- HMAC-SHA256

Oracle Identity Federation supports these session association types:

- No encryption
- Diffie-Hellman SHA1
- Diffie-Hellman SHA256

Discovery

Oracle Identity Federation supports:

- metadata publishing for OpenID Provider (OP) and Relying Party (RP)
- XRDS discovery of OPs by the RP

RP endpoint validation is implemented to defend against URL redirectors spoofing RPs to capture OpenID assertions. The RP endpoint validation by the OP is implemented by:

- XRDS discovery of the RP, checking that the endpoint is listed in the XRDS documented hosted at the realm URL
- ·Verification using the Realm as described by the OpenID specifications. The realm can either be a URI or a matching domain (*.foo.com for example). At verification, the server extracts the domain from the realm and attempts to match it to the RP endpoint. A valid domain used for verification needs to contain at least two elements (.foo.com is acceptable while .com is not). Additionally, the server provides a way to specify which domain should be discarded (for example *.co.uk).

Account Linking

Oracle Identity Federation supports persistent account linking between OpenID claimed ID and local user account by using a federation data store.

Assertions

Oracle Identity Federation supports assertion-to-user record mapping on the SP side.

Pseudonyms

Oracle Identity Federation uses opaque, persistent name identifiers in OpenID assertions. This means that:

- a federation data store is required on the IdP side
- a federation data store is optional on the SP side: if the SP is configured to use Federated Identity to map the assertion to a user, then the federation data store is required.

Profiles and Extensions

Table 2–3 shows the protocol profiles and extensions that Oracle Identity Federation supports.

Table 2–3 Oracle Identity Federation Profiles and Extensions for OpenID 2.0

Profile/Extension	Notes
Attribute Exchange (AX) extension	v. 1.0 supported
Provider Authentication Policy Extension (PAPE)	v. 1.0 supported
ICAM profile	Support for No Personal Identification Information, ·Private Personal Identifier, ·GSA Profile for OpenID, and ·NIST authentication levels

2.2.2 Choosing a Profile

This section describes considerations to keep in mind when selecting a profile to implement for the selected protocol:

- Under the SAML protocols, you can specify whether providers should exchange Oracle Identity Federation assertions using the artifact profile or the POST profile. These profiles represent different methods for secure exchange of assertions.
- Under the OpenID protocol, you can select the ICAM profile, AX extension, or PAPE extension.

This section discusses:

- Using the Artifact Profile
- Using the POST Profile
- SAML Security Considerations
- Using the SAML Attribute Sharing Profile
- Using the WS-Federation Logout Profile
- Using OpenID Profiles and Extensions

2.2.2.1 Using the Artifact Profile

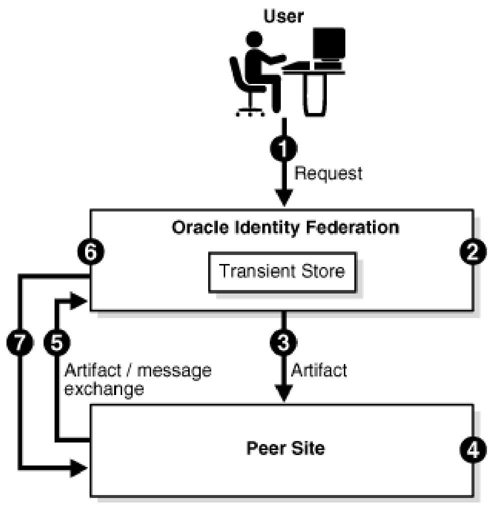
Here are some items to keep in mind when considering the artifact profile:

- The artifact profile is less resource-intensive than the POST profile because the latter uses XML signatures.
- The identity provider's SAML components must reside in the DMZ.

Artifact Profile Request Processing

Figure 2–2 shows the process by which requests are processed under the artifact profile.

Figure 2–2 Artifact Profile Processing Flow



The processing flow takes this path:

1. A user performs a request at Oracle Identity Federation (acting as the IdP).
2. Oracle Identity Federation authenticates the user and creates an artifact which includes an identifier for the IdP that is known to the SP.

   The message to be sent is stored in a repository at the server, with the artifact as a key for retrieval.

   Note: Depending on the installation, the repository may reside either in memory or in a relational database. When using replicated Oracle Identity Federation servers for high availability, the repository must reside in a database.

   ---

3. The server redirects the user to the peer site with the artifact. The artifact profile is used to carry the message.
4. The peer site decodes the artifact and deduces that Oracle Identity Federation is the originating site.
5. The peer site contacts the IdP Oracle Identity Federation, sends the artifact and asks the server to dereference it.
6. Oracle Identity Federation retrieves the message from the repository using the artifact.
7. Oracle Identity Federation sends the message to the peer site for processing.

Note: This scenario illustrates IdP-initiated single sign-on. When the request is SP-initiated, the user directly requests the resource at the service provider.

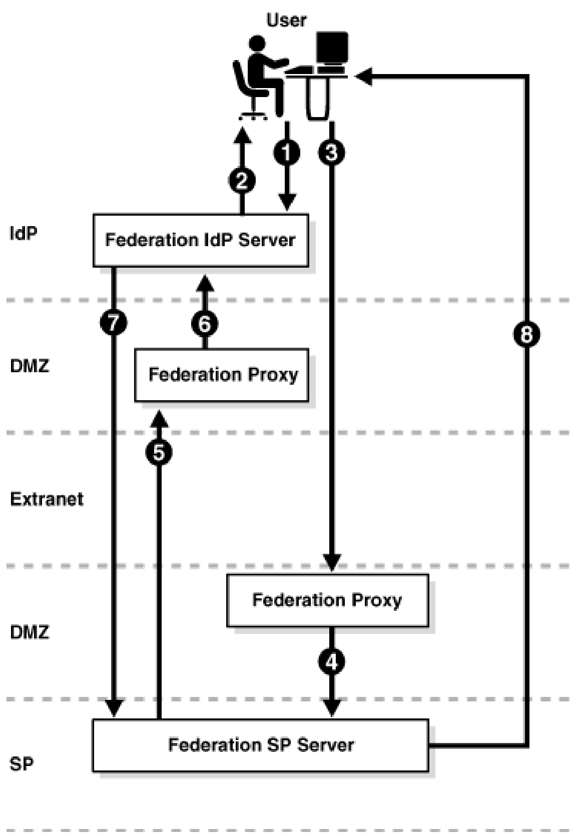
In contrast with user entries and user federation records, artifact objects are considered as transient data. Because of its transient status, the artifact has a limited lifetime and will be removed from the repository after a certain time.

Artifact Profile With Proxy

As shown in Figure 2–3, you can configure Oracle Identity Federation with proxies for IdP and SP servers when using the artifact profile. In this secure environment, the proxies are located within the DMZ.

> **See Also:** For more information about setting up a proxy server for Oracle Identity Federation, see Appendix B, "Using Oracle HTTP Server as a Proxy for Oracle Identity Federation".

Figure 2–3 Artifact Profile Processing with Proxy



The processing flow is as follows:

1. A user issues a request at the Oracle Identity Federation IdP server.
2. Oracle Identity Federation authenticates the user and creates an artifact which includes a short IdP server identifier. The server redirects the user with the artifact to the receiver service on the SP proxy server.
3. The user's browser sends the request containing the artifact to the URL of the service provider's receiver service, which is located on the proxy in the SP's DMZ.
4. The proxy forwards the request to the Oracle Identity Federation SP server.
5. The SP contacts the IdP's responder service, which is located on the proxy in the IdP's DMZ, sends the artifact, and asks the IdP to dereference it.

6. The proxy forwards the request to the IdP.
7. The IdP retrieves the message from the repository using the artifact, and sends it to the SP.
8. The SP server creates a user session and redirects the user's browser to the desired resource.

For testing purposes, you can configure the peer providers to communicate over open ports. Secure SSL ports are recommended for production, however, and the peer IdP and SP administrators must have exchanged and installed each other's CA certificates. These certificates are used to encrypt and decrypt requests and responses exchanged between the respective federation servers

2.2.2.2 Using the POST Profile

With the SAML POST profile, the identity provider sends the full assertion to the service provider over HTTPS. While testing, you may wish to configure Oracle Identity Federation without using a proxy.

Note: The assertion can be sent over HTTP as well. However, it is highly recommend that you always use HTTPS in production environments to ensure the security of the interaction.

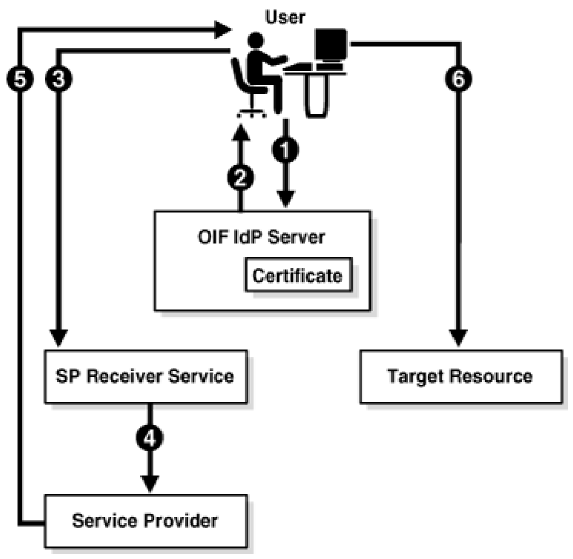
Here are some items to keep in mind when considering the POST profile:

- The POST profile does not require putting your IdP's SAML components in a DMZ.
- The SAML components can be placed behind a fire wall.
- The POST profile requires the use of XML signatures, and signing and verifying signatures is resource-intensive.
- If you plan to send or receive large numbers of requests and responses, consider Section 2.6, "Sizing Guidelines" for performance tips.

POST Profile Request Processing

Figure 2–4 shows the process by which requests are processed under the POST profile:

Figure 2–4 POST Profile Processing



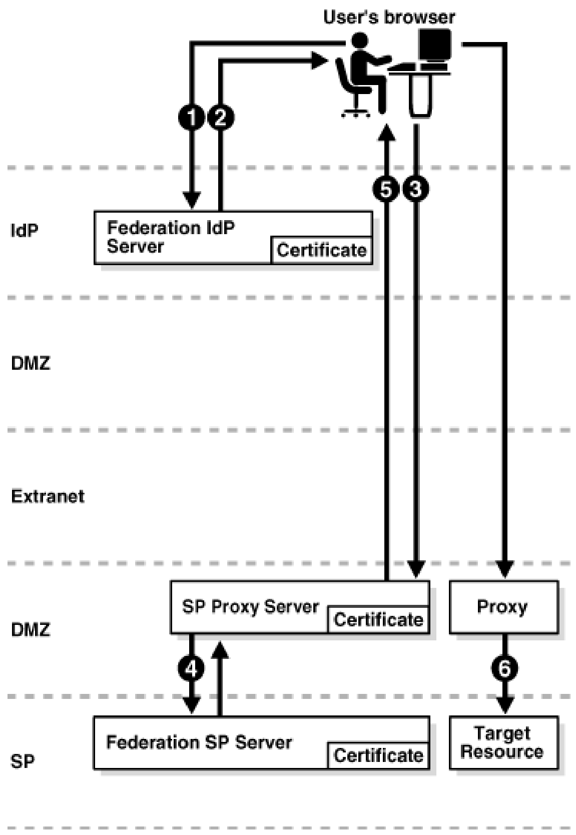
The processing flow is as follows:

1. The user initiates a request, and must be authenticated before the request can be processed.
2. Oracle Identity Federation - acting as the identity provider - authenticates the user and returns an HTML form containing a response, which consists of an identity assertion and the URL of the service provider. The response is signed using the Oracle Identity Federation IdP's private signing key.
3. The browser posts this form to the URL of the service provider's receiver service. The receiver service verifies the signed response using the IdP's public certificate associated with its signing key.
4. The service provider extracts the assertion, and creates a user session for the assertion.
5. The service provider sends the user's browser a redirect to the requested resource.
6. The user's browser sends a request to the target resource over the user session created by the service provider.

POST Profile With Proxy

In a secure deployment, the POST profile sends the full assertion to the service provider over SSL. The IdP and SP are configured to communicate over HTTPS through their SSL ports. Figure 2–5 illustrates this preferred approach for using the POST profile in production, with Oracle Identity Federation serving as the SP in the DMZ:

Figure 2–5 POST Profile with a Proxy



> **See Also:** For more information about setting up a proxy server for Oracle Identity Federation, see Appendix B, "Using Oracle HTTP Server as a Proxy for Oracle Identity Federation".

The processing flow is as follows:

1. With Oracle Identity Federation acting as the IdP, the user requests a resource. The SP, an Oracle Identity Federation server, is accessed through a proxy server located in the DMZ.
2. The IdP server authenticates the user and responds with an HTML form that contains an assertion and the URL of the target resource.

   The response is signed using the Oracle Identity Federation IdP's private signing key.
3. The user's browser posts the form to the SP's proxy receiver service URL.
4. The proxy forwards the form to the SP's receiver service.
5. The Oracle Identity Federation SP verifies the signature, extracts the assertion, creates a user session for the assertion, and sends the user's browser a redirect to the resource.
6. The browser conveys the request to the target resource over the new user session. The request may be handled by an additional proxy located in the service provider DMZ.

2.2.2.3 SAML Security Considerations

SAML provides numerous security features that you can use to ensure privacy, integrity, authenticity, and confidentiality of the SAML messages and the message exchanges.

This section provides a brief summary of message security considerations. For a detailed analysis of the security risks and countermeasures, refer to the OASIS SAML Security Considerations specification, titled *Security and Privacy Considerations for OASIS SAML V2.0*, at:

http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf

Oracle Identity Federation supports the full set of security technologies and techniques available for use in a SAML deployment. These include

- SSL/TLS for peer authentication and secure communications
- XML-SIG for message-level integrity and authentication
  - With SHA-1 for signature generation
  - With SHA-1 or SHA-256 for signature verification
- XML-ENC for message-level confidentiality

Oracle recommends that secure SSL/TLS channels be used for all SAML message flows in addition to message level security. All communications between an identity provider and service provider must use bilateral authentication (client and server certificates).

SAML profiles provide specific recommendations on how to securely use SAML assertions and request-response messages in communications protocols. Here are the security requirements for the SAML SSO Artifact and POST profiles.

Secure communication using the SAML Artifact Profile

Secure communication using the SAML artifact profile requires the following:

1. SSL is required for redirection from the IdP to the user's browser, and for redirection from the browser to the SP.
2. The SOAP channels used by the IdP and SP to communicate directly must be protected either by using SSL or by using HTTP Basic Authentication.

> **Note:** It is also possible for the SP to use HTTP basic authentication with a username and password known to the IdP.

Secure communication using the SAML POST Profile

Secure communication using the SAML POST profile requires the following:

1. Secure HTTP (HTTPS) is required to transmit a user request from a browser to the service provider.
2. The identity provider must use an XML signature to sign responses it sends to a service provider.
3. The service provider must verify the XML signature on the response.

2.2.2.4 Using the SAML Attribute Sharing Profile

The SAML attribute sharing profile is used by service providers to authenticate users by means of SSL client X.509 certificates rather than SAML assertions, when additional user attributes are needed to provide authorization of resource requests.

Oracle Identity Federation provides the attribute sharing profile for use with Oracle Access Manager to enable interoperation with SAML implementations at peer sites. For details about components and their respective roles, and how to configure Oracle Identity Federation and Oracle Access Manager, see Section 5.6.4.3, "Configuring an Oracle Access Manager Policy using Attribute Sharing".

2.2.2.5 Using the WS-Federation Logout Profile

WS-Federation can be used to sign into one or more service providers using an identity provider that performs the actual authentication.

To log out, the user clicks on a link at the IdP site that initiates a WS-Federation signout. Using a session cookie, Oracle Identity Federation has kept track of each SP to which the user signed on. The server returns an HTML signout page to the user's browser. Each SP processes the signout cleanup to sign out the session created for Oracle Identity Federation.

2.2.2.6 Using OpenID Profiles and Extensions

This section describes Oracle Identity Federation support for different OpenID profiles and extensions.

Attribute Exchange (AX)

AX is an OpenID 2.0 extension allowing user attributes to be requested and returned. OIF supports AX version 1.0.

Support on the IdP includes the following:

- Profile support is enabled on the IdP, but for each SP, you must indicate whether attributes should be sent
- Attribute definition is achieved through the existing screen on the SP Partner specific page

Support on the SP includes the following:

- Attribute definition is achieved through the existing screen on the IdP Partner specific page
- In the attribute definition page, you can specify which attributes to request from the IdP when performing the SSO protocol.
- A custom SP engine or a pre-processing engine can dictate at run-time which attributes must be requested from the IdP when performing the SSO protocol.

Provider Authentication Policy Extension (PAPE)

PAPE is an OpenID 2.0 extension allowing RPs to request specific authentication type/strength, including Levels of Assurance. Oracle Identity Federation supports PAPE version 1.0.

Support on the IdP/OP includes the following:

- The IdP publishes in the XRDS document whether or not the PAPE extension is enabled.

- If enabled, the IdP includes the authentication mechanism used to authenticate the user in the response to the SP.

Support on the SP/RP includes the following:

- If the IdP supports PAPE, and if configured to request a specific authentication mechanism, the SP indicates the mechanism to use to authenticate the user at the IdP.

Note: The Oracle Identity Federation authentication mechanism is translated to OpenID authentication methods.

US Government Federal Identity, Credentialing and Access Management (ICAM) Profile

Oracle Identity Federation supports these privacy policy and security requirements for the US Government for OpenID 2.0 deployments:

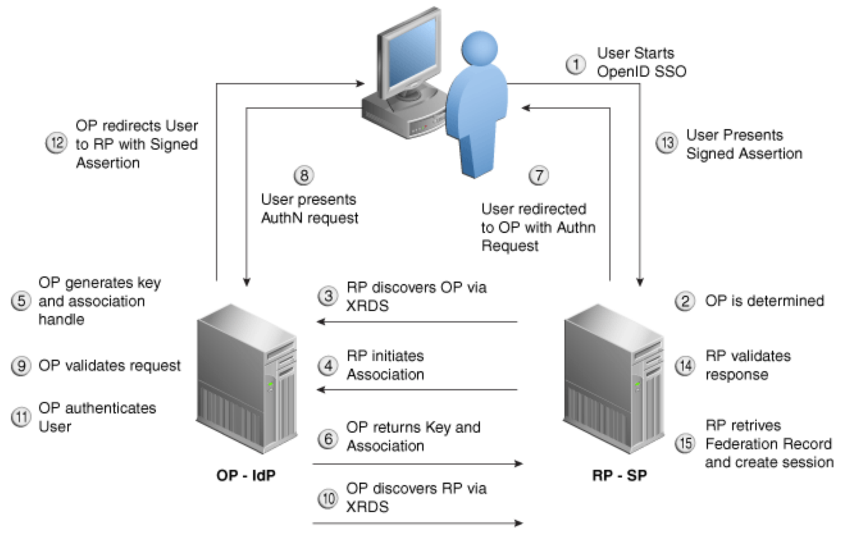
- No Personal Identification Information referenced by the ICAM OpenID 2.0 Profile URI. When enabled and specified in the protocol exchange, the IdP cannot include any personal information in the response to the SP.
- Private Personal Identifier referenced by the OASIS Private Personal Identifier URI. When enabled and specified in the protocol exchange, the IdP must return an opaque ClaimedID specific to the RP.
- GSA Profile for OpenID referenced by the ICAM OpenID 2.0 Profile URI. When enabled and specified in the protocol exchange, the IdP must follow GSA profile rules when performing the OpenID SSO protocol.
- NIST authentication levels referenced by the http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf URI. When enabled, the IdP includes the NIST Level Of Assurance information in the response to the SP.

Note: While these profiles can be enabled on Oracle Identity Federation, you must ensure that the federation server complies with the requirements.

OpenID Profile Request Processing

Figure 2–6 shows the request processing under the OpenID profile:

Figure 2–6 OpenID Processing Flow



1 User Starts OpenID SSO
2 OP is determined
3 RP discovers OP via XRDS
4 RP initiates Association
5 OP generates key and association handle
6 OP returns Key and Association
7 User redirected to OP with Authn Request
8 User presents AuthN request
9 OP validates request
10 OP discovers RP via XRDS
11 OP authenticates User
12 OP redirects User to RP with Signed Assertion
13 User Presents Signed Assertion
14 RP validates response
15 RP retrives Federation Record and create session
OP - IdP
RP - SP

2.3 Authentication Engines

Many Oracle Identity Federation features require the user to be authenticated. Such operations include:

- IdP protocol operations such as single sign-on, federation creation, federation termination, and NameID registration
- SP protocol operations such as federation creation, federation termination, and NameID registration

To gain a perspective on how authentication is effected, we can think of the federation server as comprising these distinct modules:

1. Oracle Identity Federation provides support for WS-Federation, Liberty 1.x, SAML 1.0/1.1, SAML 2.0, and OpenID protocols.
2. An authentication module provides support for user authentication and integration with IdM solutions.

To support these operations, Oracle Access Manager provides a range of identity administration functions including Web single sign-on, user self-service and registration, policy management, and delegated administration.

In this section we look at the authentication flows these modules enable in different configurations:

- Engines in Oracle Identity Federation
- Authenticating with a Repository
- Authenticating with an IdM Solution in IdP Mode
- Propagating Authentication State to Oracle Access Manager in SP Mode
- Propagating Authentication State to Oracle Single Sign-On in SP Mode
- HTTP Basic Authentication

2.3.1 Engines in Oracle Identity Federation

Oracle Identity Federation interacts with two distinct modules when performing User Federation operations:

- The authentication engine acts as a local authentication mechanism. In this mode, the authentication module can authenticate locally with available authentication systems.

  Oracle Identity Federation conveys authentication requests to the authentication module. Depending on the deployment, the authentication module may interact directly with RDBMS or LDAP repositories, or it may delegate authentication to an IdM solution such as Oracle Single Sign-On.

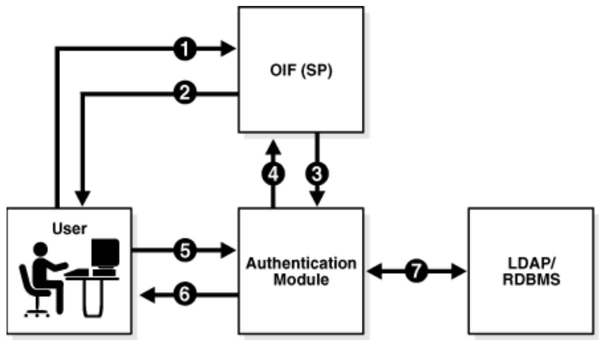
- The Oracle Identity Federation SP integration engine acts to propagate the authentication state. In this mode, Oracle Identity Federation, as a service provider, uses federation protocols to have the user authenticated at a peer identity provider. Oracle Identity Federation then forwards the user to the authentication module, which propagates and creates an authenticated user session in the deployed IdM solution at the SP. In turn, this enables access to the requested protected resource.

2.3.2 Authenticating with a Repository

In this deployment, the authentication module interacts directly with a number of repositories and IdM solutions to enable Oracle Identity Federation to locally authenticate the user:

- an RDBMS repository
- an LDAP repository

Figure 2–7 Authenticating with a Repository in IdP Mode



The flow for a local authentication involving such a deployment is as follows:

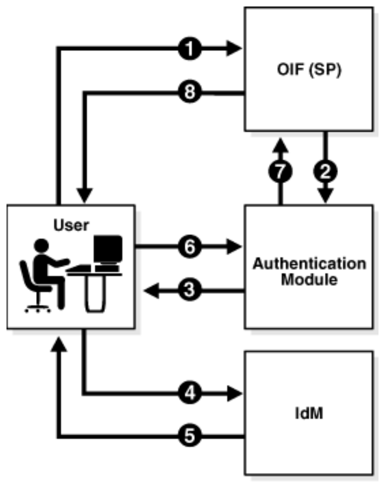
- The user accesses Oracle Identity Federation (Step 1).
- Oracle Identity Federation forwards the user to the authentication module for local authentication (Step 3).
- The user enters credentials (Step 5), when
- the authentication module prompts the user for credentials (Step 6)
- The authentication module interacts with the repository to authenticate the user (Step 7).
- The authentication module forwards the user to Oracle Identity Federation with the user's identification (Steps 6,1).

- Oracle Identity Federation communicates with the authenticated user (Step 2).

2.3.3 Authenticating with an IdM Solution in IdP Mode

In this deployment, the authentication module delegates authentication to the Oracle Single Sign-On IdM solution or Oracle Access Manager to enable Oracle Identity Federation to authenticate in IdP mode.

Figure 2–8 Authenticating with an IdM Solution in IdP Mode



The flow for a local authentication involving an IdM deployment is as follows:

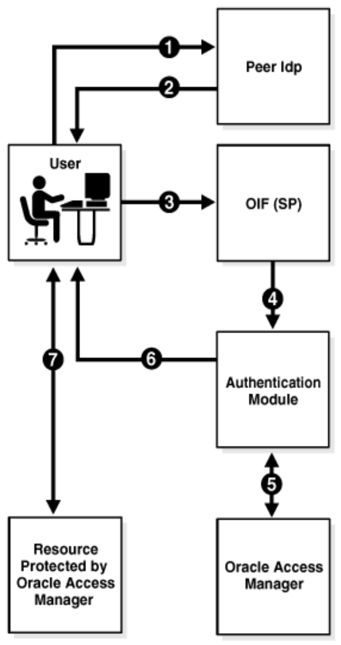
- The user accesses Oracle Identity Federation (Step 1).
- Oracle Identity Federation forwards the user to the authentication module for local authentication (Step 2).
- The authentication module redirects the user to the IdM server for authentication (Steps 3,4).
- The IdM server authenticates the user and redirects the user back to the authentication module (Steps 5,6).
- The authentication module forwards the user to Oracle Identity Federation with the user's identification (Step 7).
- Oracle Identity Federation communicates with the authenticated user (Step 8).

2.3.4 Propagating Authentication State to Oracle Access Manager in SP Mode

In this mode, Oracle Identity Federation uses the federation protocols to identify a user, and requests the authentication module to create an authenticated session at Oracle Access Manager so that the user can access the requested resource, which is protected by WebGate (for an Oracle Access Manager IdM deployment).

The request originates at a peer IdP, and Oracle Identity Federation authenticates in SP mode.

Figure 2–9 Authenticating with Oracle Access Manager in SP Mode



The flow for authenticating a user at a peer provider with Oracle Access Manager is as follows:

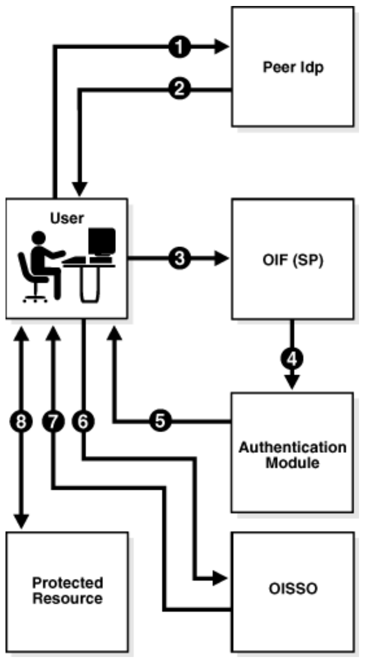
- The user is at the peer IdP (Step 1).
- The IdP redirects the user to Oracle Identity Federation (as SP) with an authentication assertion (Steps 2,3).
- Oracle Identity Federation processes the assertion, creates a local Oracle Identity Federation session, and forwards the user to the authentication module with the identification (Step 4).
- The authentication module interacts with Oracle Access Manager to create an Oracle Access Manager authenticated session (Step 5).
- The authentication module redirects the user to the protected resource (Step 6).
- WebGate Web Agent grant the user access to the protected resource (Step 7).

2.3.5 Propagating Authentication State to Oracle Single Sign-On in SP Mode

In this mode, Oracle Identity Federation uses the federation protocols to identify a user, and requests the authentication module to create an authenticated session at Oracle Single Sign-On so that the user can access the requested resource, which is protected by `mod_osso`.

The request originates at a peer IdP, and Oracle Identity Federation authenticates in SP mode.

Figure 2–10 Authenticating with Oracle Single Sign-On in SP Mode



The flow for authenticating a user at a peer provider with Oracle Single Sign-On is as follows:

- The user is at the peer IdP (Step 1).
- The IdP redirects the user to Oracle Identity Federation (as SP) with an authentication assertion (Steps 2,3).
- Oracle Identity Federation processes the assertion, creates a local Oracle Identity Federation session, and forwards the user to the authentication module with the identification (Step 4).
- The authentication module redirects the user to Oracle Single Sign-On with the user identification (Steps 5,6).
- Oracle Single Sign-On creates a local authenticated session and grants access to the resource protected by `mod_osso` (Steps 7,8).

2.3.6 HTTP Basic Authentication

Oracle Identity Federation can be configured to accept HTTP basic credentials without requiring an identity and access management system. This corresponds to using the JAAS authentication engine.

2.4 Data Repositories

This section describes installation requirements to enable Oracle Identity Federation to work with data stores. It contains these topics:

- Federation Data Store
- User Data Store
- Session and Message Data Stores
- Configuration Data Store

2.4.1 Federation Data Store

You must select a data repository for the persistent federation data store. Oracle Identity Federation works with industry-standard LDAP repositories including:

- Oracle Internet Directory
- Sun Java System Directory Server
- Microsoft Active Directory
- IBM Tivoli

It also supports XML stores, databases, and a None option (no repository) for SAML and WS-Federation using non-opaque name identifiers such as e-mail address, X.509 DN, Kerberos, or Windows Name Identifier.

Connection Information

Collect the following information about the repository prior to installing Oracle Identity Federation:

- The Connection URL (space-delimited list of LDAP server URLs - hostname and port)
- The Bind DN

  This is the DN used by the Oracle Identity Federation server to connect to the LDAP server. For example:

  ```
  cn=fedid,dc=mycompany,dc=com
  ```

- Password
- The User Federation Record Context

  This is the node under which all federation records for this Oracle Identity Federation server will be stored.

- The LDAP Container Object Class

  This is the type of User Federation Record Context that Oracle Identity Federation should use when creating the LDAP container, if it does not exist already. If that field is empty, its value will be set to `applicationprocess`. For Microsoft Active Directory this field has to be set, to container for example. The appropriate setting for this field depends on the User Federation Record Context being used. (User Federation Record Context is described later in this section).

  Here are examples of the LDAP Container Object Class for different types of directory servers:

  – Oracle Internet Directory: *empty*

  – Oracle Directory Server Enterprise Edition: *empty*

  – Microsoft Active Directory: `container`

- Unique Federation ID Attribute

  This is the LDAP attribute to be used to uniquely identify a federation record. This attribute should be defined in the LDAP Object Class of the Federation Record type, or in its top parent. If it is empty, the default Federation ID attribute will be used as the DN of the Federation Record.

  Here are examples of the Unique Federation ID attribute for different types of directory servers:

  - Oracle Internet Directory: *empty*
  - Oracle Directory Server Enterprise Edition: *empty*
  - Microsoft Active Directory: *empty*
- Maximum Connections. This is the maximum number of concurrent connections made by Oracle Identity Federation to the LDAP server.
- Connection Wait Timeout. This is the maximum number in seconds to wait until a connection is available, when the maximum number of connections opened by Oracle Identity Federation to the LDAP server has been reached.

Relationship of User Federation Record Context and LDAP Container Object Class

The User Federation Record Context and LDAP Container Object Class must be compatible. In the User Federation Record Context, the administrator specifies the DN of the container where the federation records will be stored. That DN will contain the parent of the container that must already exist (for example `dc=us,dc=oracle,dc=com`), and an attribute of the Federation Record Context that is part of its object class (for example, `cn=orclfed`). An example of such DN would be `cn=orclfed,dc=us,dc=oracle,dc=com`.

The requirement for that example is that `cn` must be an attribute of the Object Class set in the LDAP Container Object Class field (or the `applicationprocess` object class if not set).

If the administrator chooses to have the DN of the Federation Record Context like `ou=fed,dc=us,dc=oracle,dc=com`, she must set the LDAP Container Object Class field to an object class that has `ou` as an attribute, like `organizationalUnit`.

To summarize, the User Federation Record Context references the LDAP container entry under which federation records are stored, and the LDAP Container's attribute used in the DN must be defined in the LDAP Container Object Class used. For example, if DN is `ou=fed,dc=us,dc=oracle,dc=com`, then the LDAP Container Object Class must define the `ou` attribute; if DN is `cn=fed,dc=us,dc=oracle,dc=com`, then the LDAP Container Object Class must define the `cn` attribute.

A Note About the LDAP Schema

The LDAP schema needs to be upgraded to include the attributes and object classes defined by Oracle Identity Federation, in order for the federation server to create records in the LDAP server.

Upgrade the LDAP schema either at installation time (with the Advanced Installation mode), or after installation.

Upgrade Schema at Installation

To perform the upgrade at installation time, take these steps:

1. Choose the Advanced Installation mode.
2. On the "Select Configuration Options" page, check the "Federation Data in LDAP Server" box. This indicates that the federation records will be stored in an LDAP server whose schema must be upgraded.
3. On the "Specify Federation Data Store" page, enter the LDAP connection information. The schema is then upgraded as part of the installation process.

Post-Installation Schema Upgrade

To perform the upgrade post-installation, note that the Oracle Identity Federation installation includes `LDIF` files that you can execute using the `ldapmodify` tool to upgrade the schema of an LDAP server.

The `LDIF` file to use depends on the type of LDAP server used:

- `$Oracle_Home/fed/setup/ldap/userFedSchemaOid.ldif` if you use Oracle Internet Directory
- `$Oracle_Home/fed/setup/ldap/userFedSchemaSunOne5.ldif` if you use the Oracle Directory Server Enterprise Edition 5.x
- `$Oracle_Home/fed/setup/ldap/userFedSchemaSunOne6.ldif` if you use the Oracle Directory Server Enterprise Edition 6.x
- `$Oracle_Home/fed/setup/ldap/userFedSchemaAD.ldif` if you use Microsoft Active Directory Server. In this case, you must edit the `LDIF` file to replace the string `%DOMAIN_DN%` with your active directory domain suffix.

  An example suffix is `dc=mydomain,dc=mycompany,dc=com`.
- `$Oracle_Home/fed/setup/ldap/userFedSchemaTivoli.ldif` if you use the IBM Tivoli Directory Server (IBM TDS) 6.0

Using `ldapmodify`, you can upgrade the LDAP schema with the LDIF file. For example:

```
ldapmodify -c -D BIND_DN_USERNAME
   -w PASSWORD
   -f $Oracle_Home/fed/setup/ldap/userFedSchemaOid.ldif
   -h LDAP_HOSTNAME -p LDAP_PORT -x
```

2.4.2 User Data Store

You must select a data repository for the user data store. Oracle Identity Federation works with industry-standard repositories including:

- LDAP (Oracle Internet Directory, Sun Java System Directory Server, and Microsoft Active Directory)
- RDBMS

The role played by the data repository depends on whether Oracle Identity Federation will be configured as an identity provider (IdP) or a service provider (SP):

- As an IdP, Oracle Identity Federation uses the repository to verify user identities and to build protocol assertions.
- As an SP:
  - Oracle Identity Federation uses the repository to map information in received assertions to user identities at the destination, and subsequently to authorize users for access to protected resources.
  - When creating a new federation, Oracle Identity Federation uses the repository to identify the user and link the new federation to that user's account.

Connection Information for LDAP Repositories

Collect the following information about the repository prior to installing Oracle Identity Federation:

- Connection URL - space delimited list of LDAP URLs

- Bind DN
- Password
- User ID Attribute - the attribute name to use to map users during lookups or authentication procedures

  Here are examples of the User ID Attribute for different types of directory servers:

  – Oracle Internet Directory: `uid`

  – Oracle Directory Server Enterprise Edition: `uid`

  – Microsoft Active Directory: `sAMAccountName`
- User Description Attribute

  This field references the user attribute to use as a human readable federation owner identifier. This information will be stored in the federation record.

  Here are examples of the User Description Attribute for different types of directory servers:

  – Oracle Internet Directory: `uid`

  – Oracle Directory Server Enterprise Edition: `uid`

  – Microsoft Active Directory: `sAMAccountName`
- Person Object Class - the LDAP object class representing a user in the LDAP server

  Here are examples of the Person Object Class for different types of directory servers:

  – Oracle Internet Directory: `inetOrgPerson`

  – Oracle Directory Server Enterprise Edition: `inetOrgPerson`

  – Microsoft Active Directory: `user`
- Base DN - the node under which LDAP user search will be performed. For example:

  ```
  dc=us,dc=oracle,dc=com
  ```
- Maximum Connections - the maximum number of concurrent connections made by Oracle Identity Federation to the LDAP server
- Connection Wait Timeout - the maximum number in seconds to wait until a connection is available, when the maximum number of connections opened by Oracle Identity Federation to the LDAP server has been reached

Connection Information for RDBMS Repositories

Collect the following information about the repository prior to installing Oracle Identity Federation:

- JNDI Name - references the data source configured in Oracle WebLogic Server pointing to the RDBMS to use to authenticate/locate users. You must define this data source after Oracle Identity Federation installation, prior to authenticating any users.
- Login Table - the RDBMS table containing the user information used for authentication and lookups

- User ID Column - the RDBMS column in the login table containing the user identifiers
- User Description Attribute - references the user attribute to use as a human readable federation owner identifier. This information will be stored in the federation record.

2.4.3 Session and Message Data Stores

Oracle Identity Federation also maintains transient session and message data stores for federation session/protocol state. This data can be stored in either in-memory tables or a relational database.

RDBMS session and message data stores are required for high-availability and clustering support.

2.4.4 Configuration Data Store

Configuration data for Oracle Identity Federation can be stored in either XML files or a relational database.

An RDBMS configuration data store is required for high-availability and clustering support.

2.5 Installation Requirements

This section explains installation requirements.

Note: Liberty 1.x support is deprecated.

2.5.1 Required Components

Oracle Identity Federation requires the following components:

- Java 2 SDK, Standard Edition (J2SE), Version 1.4.2 (bundled with the installation)
- Oracle WebLogic Server
- A user identity data store. This is typically an LDAP directory, but can optionally be a database store.
- One of these repositories for the user federation data store:
  - Oracle Internet Directory
  - Microsoft Active Directory
  - Sun Java System Directory Server

  Note: A user federation data store is not absolutely required for Oracle Identity Federation in all cases: it is required for Liberty 1.x and SAML 2.0 opaque persistent identifiers, but is optional for SAML 1.x, WS-Federation, and SAML 2.0 non-opaque identifiers (such as email address, subject DN, and so on).

- One of these versions of Oracle Database for the RDBMS transient data store:
  - Oracle Database 10.2.0.4 or higher

  - Oracle Database 11.1.0.7 or higher
  - Oracle Database 11.2.x

Note: Check the certification matrix for the most current version information.

- Oracle HTTP Server for proxy implementation; this is the only proxy server supported by Oracle Identity Federation, and is bundled with the installation.

2.6 Sizing Guidelines

When planning to deploy a federated identity system that leverages Oracle Identity Federation, it is critical to understand the performance considerations, choices, and trade-offs involved in the architecture.

This section considers various factors that have an impact on performance in a federated environment, and provides some guidelines to help you assess hardware requirements for a production system with a standalone Oracle Identity Federation server. The following topics are included:

- Deployment and Architecture Considerations
- Typical Deployment Scenario
- Reference Server Footprint
- Topology

2.6.1 Deployment and Architecture Considerations

Before deploying Oracle Identity Federation, you must define the architecture and role that Oracle Identity Federation will play in a federated authentication setting. Here are some decisions that you must make:

- Which federation specifications will be used with various trusted partners? Choices include:
  - SAML 2.0. With additional flows in comparison to SAML 1.x, performance considerations may play a greater role.
  - SAML 1.0 and 1.1
  - WS-Federation
- What profiles will you use to federate with your partners? Options include Browser POST or Artifact profile, WS-Federation Passive Requestor profile, attribute sharing, and others.
- Which transport security protocols and certificates will be used? Will the assertions be signed?
- What roles will Oracle Identity Federation be playing? Options are:
  - Identity Provider (IdP), also referred to as a source domain
  - Service Provider (SP), also referred to as a destination domain
  - Both IdP and SP
- What type (and what vendor's) authoritative identity repositories will be installed?

Note: Oracle Identity Federation provides an integration framework that enables you to create lightweight federation endpoints without requiring an access management system.

- Will you install a proxy server with Oracle Identity Federation? If so, take into account where the Oracle Identity Federation and proxy servers will reside - for example, in the DMZ or behind a firewall.
- How will the architecture provide high availability scenarios? Specifically:
  - Whether you want to support cold failover clusters leveraging the Oracle Application Server High Availability topologies
  - Whether you want to set up a common assertion store database to make assertion data available to more than one federation server in a load-balancing and failover configuration

The overall throughput and performance of Oracle Identity Federation can depend on a number of factors, such as:

- Which profiles are supported (for example, Artifact or POST)
- Security features in use (using certificates, digitally signing and/or encrypting assertions)
- Use of individual components involved in processing a transaction, such as fire walls, proxy servers, LDAP directories, databases, and IAM systems

The subsequent subsections provide more detail on these topics:

- Profiles
- Repositories
- Transient (Session and Message) Storage
- Security for Assertions
- Connection Tuning
- High Availability
- Tuning Servers
- HTTP Session Persistence
- Impact of Additional Security

2.6.1.1 Profiles

The SAML specification supports a number of profiles, with the two primary deployed profiles being the SAML Browser POST and Artifact profiles. In general, using the SAML Browser POST profile is more performance-friendly than the Artifact profile, as the POST profile requires fewer round trips between the IdP and SP. However, there is a potential security trade-off given that the Artifact profile is, in general, a more secure method of exchanging SAML assertions.

2.6.1.2 Repositories

When working with LDAP directories, RDBMS, and back end IAM systems, it is important to pay attention to the transaction processing speed of the component in question, since this can affect the performance of your production environment. Note that:

- RDBMS parameters can be tuned to provide options to control database connection pool settings.
- If using Oracle Access Manager as the back-end identity and access management system, the AccessGate performance considerations apply, as do the Access Server sizing considerations outlined here:

  `http://docs.oracle.com/cd/E15217_01/doc.1014/e12490.pdf`

2.6.1.3 Transient (Session and Message) Storage

Place the transient data store in memory for improved performance.

2.6.1.4 Security for Assertions

Performance can be sensitive to the presence or absence of digital signatures/encryption in the SAML assertions. While removing these features can improve performance, it is not recommend if the IdP and SP communication takes place over the internet.

2.6.1.5 Connection Tuning

Pay attention to the proper adjustment of the maximum number of concurrent connections to:

- LDAP servers,
- the RDBMS (for transient session data and configuration), and
- remote providers (when Oracle Identity Federations interact directly with each other using the SOAP protocol)

2.6.1.6 High Availability

For greater performance and high availability, consider scaling and load-balancing multiple Oracle Identity Federation servers. Implementing a load-balancing solution provides backup and failover protection for your site.

For details, see the *Oracle Fusion Middleware High Availability Guide*.

2.6.1.7 Tuning Servers

Take into account the presence of other servers in your production environment. Specifically, consider:

- Tuning Oracle WebLogic Server and setting appropriate connection limits for Oracle Identity Federation. You can:
  - Tune Oracle WebLogic Server using typical configuration parameters such as memory used, number of processes, and so on. For details, see *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.
  - Specify the maximum number of HTTP/JDBC connections that Oracle Identity Federation uses when communicating with remote HTTP servers and RDBMS servers. For details, see *Oracle Fusion Middleware Performance and Tuning Guide*.
- Tuning the Oracle HTTP Server, which is leveraged by Oracle Identity Federation.

See Also: The following provide tuning and performance guidelines:

- Section 6.3.1, "Configuring the LDAP Inactivity Setting"
- Section 6.4.2, "Configuring the HTTP Session State Sleep/Retry Interval"
- Section 6.5.1, "Configuring RDBMS Session Cache"
- Section 6.6.1, "Storing Assertion Attributes of User Session"
- Chapter 7, "Diagnostics and Auditing"
- *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*
- *Oracle Fusion Middleware Performance and Tuning Guide*

2.6.1.8 HTTP Session Persistence

Oracle Identity Federation uses HTTP session state during request processing. To configure Oracle WebLogic Server session persistence see the chapter titled "Using Sessions and Session Persistence" in *Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle*.

By default, memory-based storage is used. If you do not allow sufficient heap size when running Oracle WebLogic Server, your server may run out of memory under heavy loads.

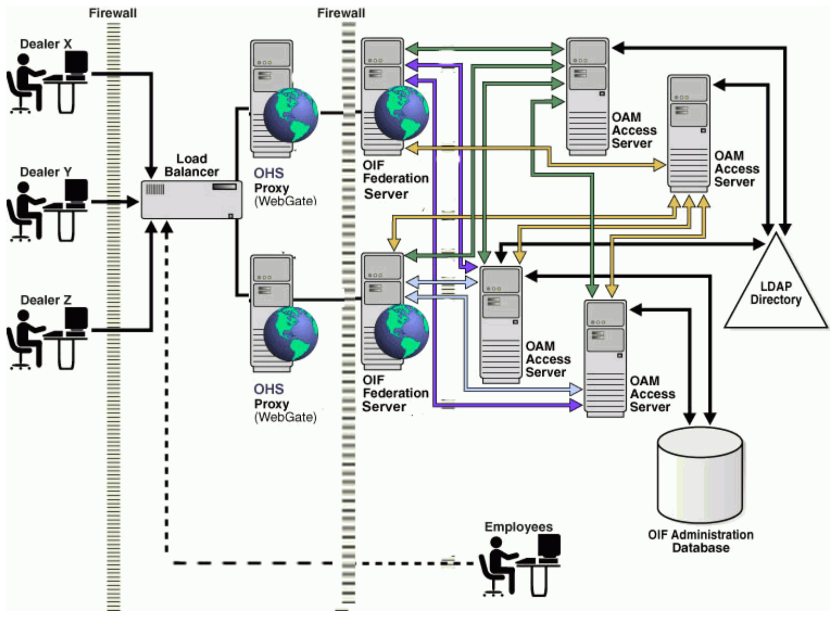
2.6.1.9 Impact of Additional Security

Introducing additional security measures, such as fire walls, proxy servers, or using SSL authentication, can add extra steps in federated transactions and therefore impact performance.

2.6.2 Typical Deployment Scenario

Figure 2–11 illustrates a typical Oracle Identity Federation deployment architecture for a service provider, where Oracle Identity Federation relies on Oracle Access Manager as the back end access management system. The diagram illustrates multiple partners coming in through the DMZ and accessing a load-balanced pair of Oracle Identity Federation Proxy Servers, which are front-ending a pair of Oracle Identity Federation servers.

Figure 2–11 A Typical Federation Deployment Architecture



2.6.3 Reference Server Footprint

The following hardware and equipment is recommended for a baseline Oracle Identity Federation deployment, for an environment supporting up to 2000 concurrent users:

- Any supported server-class machine and operating system for Oracle Identity Federation. See the certification matrix for a list of certified platforms for Oracle Identity Federation.

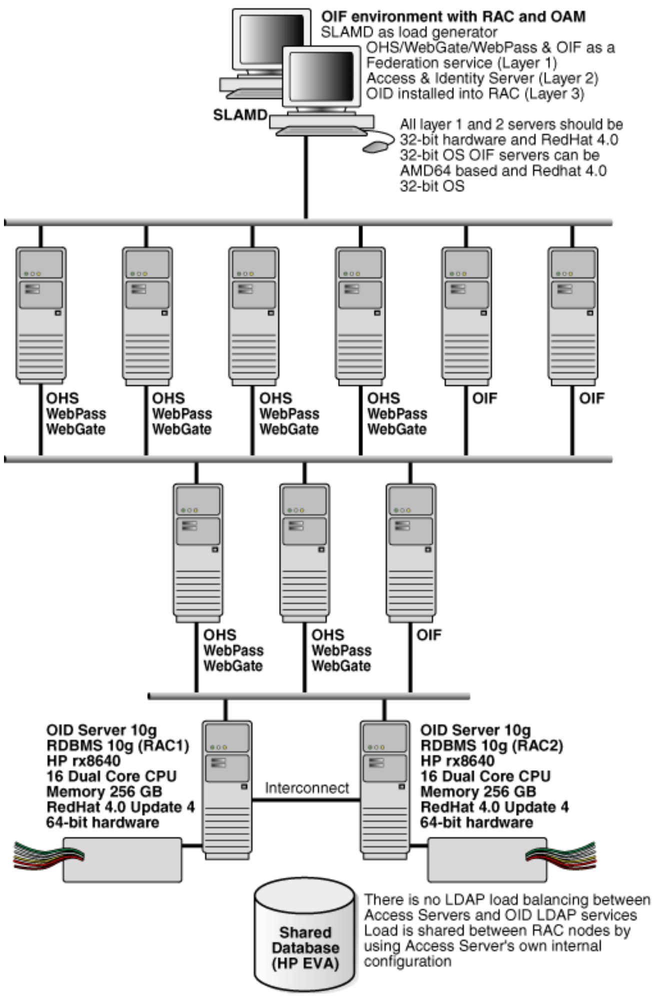
  Failover scenarios would double the number of machines required. Use a minimum of two Oracle Identity Federation servers, on separate machines, for redundancy.

- Server footprint:
  - 2-4 GB memory (4GB recommended)
  - Minimum of 2 CPUs per machine
- If a proxy server is being used, follow the vendor-specific sizing recommendations.

2.6.4 Topology

Figure 2–12 shows the recommended topology for an Oracle Identity Federation deployment in SP mode in which Oracle HTTP Server serves up a provider application that is protected by a webgate.

Figure 2–12 Sample Topology for Oracle Identity Federation


OIF environment with RAC and OAM
SLAMD as load generator
OHS/WebGate/WebPass & OIF as a Federation service (Layer 1)
Access & Identity Server (Layer 2)
OID installed into RAC (Layer 3)
SLAMD
All layer 1 and 2 servers should be 32-bit hardware and RedHat 4.0 32-bit OS OIF servers can be AMD64 based and Redhat 4.0 32-bit OS
OHS WebPass WebGate
OIF
OID Server 10g RDBMS 10g (RAC1) HP rx8640 16 Dual Core CPU Memory 256 GB RedHat 4.0 Update 4 64-bit hardware
Interconnect
OID Server 10g RDBMS 10g (RAC2) HP rx8640 16 Dual Core CPU Memory 256 GB RedHat 4.0 Update 4 64-bit hardware
Shared Database (HP EVA)
There is no LDAP load balancing between Access Servers and OID LDAP services Load is shared between RAC nodes by using Access Server's own internal configuration

2.7 Implementation Checklist

The following checklist summarizes the key items for planning an Oracle Identity Federation installation and provides the essential starting point for deployment.

Note: Liberty 1.x support is deprecated.

Table 2–4 Implementation Checklist

Planning Item	Recommended / Proposed Value	Notes
Architecture/Basic Configuration		
role played		IdP, SP, or both
protocol		Liberty 1.1, Liberty 1.2, SAML 2.0, or any combination of the three. SAML 1.0, SAML 1.1, and WS-Federation.

Table 2–4 (Cont.) Implementation Checklist

Planning Item	Recommended / Proposed Value	Notes
Repository		Specify repository for the user data and federation persistent data.
LDAP server hostname		for example, `ldap.mydomain.com`
LDAP server port number		for example, `389`
LDAP server access credentials		for example, `Bind DN = {cn=orcladmin}, Password = {mysecret}`
Base DN		for example, `dc=mydomain,dc=com`
federation record context		for example, `cn=fed,dc=mydomain,dc=com`
federation schema update[1]		This information must be provided at the time of installation.
transient data store		Specify repository for transient data: RDBMS or in-memory.
Configuration data store		Specify repository for transient data: RDBMS or File
IdP Profiles & Bindings		Use a row for each combination enabled.
SP Profiles & Bindings		Use a row for each combination enabled.
SSL Encryption		
Enabled/Disabled		
Java keystore for SSL		For information about setting up SSL, see Section 8.1, "Configuring SSL for Oracle Identity Federation".
Certificates		
signing		Specify location of PKCS #12 wallet for signing key pair.
encryption		Specify location of PKCS #12 wallet for encryption key pair.
Performance Planning		

Table 2–4 (Cont.) Implementation Checklist

Planning Item	**Recommended / Proposed Value**	**Notes**
Topology, Reference Server Footprint		For performance tips and recommendations, see *Oracle Fusion Middleware Performance and Tuning Guide*.

[1] For the federation schema update, collect the Connection URL, the Bind DN, password, User Federation Record Context, the LDAP Container Object Class (Microsoft Active Directory), and Unique Federation ID Attribute.

3 Deploying Oracle Identity Federation

This chapter describes key deployment scenarios, including integration with identity and access management systems, Web servers, and back-end data stores. It contains these topics:

- Introduction
- Deployment Scenarios
- Post-Upgrade Administration

3.1 Introduction

Oracle Identity Federation operates in a heterogeneous environment and is interoperable with a wide variety of platforms and applications. It supports multiple options for data stores and authentication providers.

To resolve deployment issues and questions, refer to Chapter 2, "Planning Oracle Identity Federation Deployment" which provides extensive background information to help you plan your deployment:

- Section 2.1, "Architecture Options" provides details about supported protocols and profiles, and what to consider when evaluating deployment options.
- Section 2.6, "Sizing Guidelines" explains performance factors and provides topology recommendations.
- Section 2.7, "Implementation Checklist"provides a deployment checklist.

The next section describes different deployment scenarios and provides step-by-step instructions for configuring Oracle Identity Federation to work with key components of the federation environment.

3.2 Deployment Scenarios

This section describes the steps needed to implement common Oracle Identity Federation deployment scenarios. It contains these sections:

- Deploying Oracle Identity Federation with Oracle HTTP Server
- Deploying Oracle Identity Federation with Oracle Single Sign-On
- Deploying Oracle Identity Federation with Oracle Access Manager 11g
- Deploying Oracle Identity Federation with Oracle Access Manager 10g
- Oracle Identity Federation/SP Authenticating to Oracle Access Manager

- Deploying Oracle Identity Federation with Oracle Directory Server Enterprise Edition
- Using the Test SP Engine

3.2.1 Deploying Oracle Identity Federation with Oracle HTTP Server

HTTP Servers are deployed in the Web tier.

Most Identity Management components can function without the Web tier, but for most enterprise deployments, the Web tier is desirable. To support enterprise level single sign-on using products such as Oracle Single Sign-On and Oracle Access Manager, the Web tier is required.

This section describes the steps needed to install and deploy Oracle Identity Federation so that it is integrated with Oracle HTTP Server (OHS).

- Install Oracle HTTP Server
- Manage the Oracle HTTP Server Instance
- Associate Oracle HTTP Server with Managed Server
- Update Oracle Identity Federation Configuration

3.2.1.1 Install Oracle HTTP Server

Note: When Oracle HTTP Server is installed, only the HTTP protocol is enabled. To enable SSL between Oracle HTTP Server and the managed server running Oracle Identity Federation, you must configure HTTPS post-install.

For details, refer to *Oracle Fusion Middleware Administrator's Guide*.

When installing the IdM suite, select Oracle HTTP Server in the **Select Components** screen. This will install Oracle HTTP Server.

After installation, issue the following command-line instruction to create the instance:

```
$AS_INST/bin/opmnctl createcomponent -componentType OHS -componentName $OHS_NAME
```

where `$AS_INST` represents the application server instance home, and `$OHS_NAME` is the name of the new OHS component.

3.2.1.2 Manage the Oracle HTTP Server Instance

The commands to start, stop, and restart Oracle HTTP Server respectively are as follows:

```
$AS_INST/bin/opmnctl startproc process-type=OHS
$AS_INST/bin/opmnctl stopproc process-type=OHS
$AS_INST/bin/opmnctl restartproc process-type=OHS
```

where `AS_INST` represents the application server instance home.

3.2.1.3 Associate Oracle HTTP Server with Managed Server

Next, take these steps to link Oracle HTTP Server to the managed server where Oracle Identity Federation is running:

1. Open `$AS_INST/config/OHS/$OHS_NAME/moduleconf/oif.conf`

where `$AS_INST` represents the application server instance home.

2. If Oracle Identity Management was installed in stand-alone mode, uncomment and set the `WebLogicHost` and `WebLogicPort` variables to reference the WebLogic managed server where Oracle Identity Federation is running (for example `myhost.us.mycorp.com` and `7499`).
3. If the Oracle Identity Management is installed in clustered mode, uncomment and set the `WebLogicCluster` variable to reference the Oracle WebLogic Server managed servers where Oracle Identity Federation is running (for example `myhost1.us.mycorp.com:7499,myhost2.us.mycorp.com:7499`).
4. Save the file and exit.
5. Restart Oracle HTTP Server.

See Also: Understanding Key Oracle Fusion Middleware Concepts in the *Oracle Fusion Middleware Administrator's Guide*.

3.2.1.4 Update Oracle Identity Federation Configuration

Next, take these steps to update the Oracle Identity Federation configuration:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Server Properties**.
3. Change the hostname to reflect the hostname configured in OHS, if they are different.
4. Change the Port/SSL Enabled and SOAP Port/SSL Enabled information to reflect the OHS configuration.
5. Save the changes.
6. Since the Oracle Identity Federation metadata has changed, redistribute the metadata to any remote partners to notify them of the changes.

See Also: Section 5.2, "Configuring Server Properties"

3.2.2 Deploying Oracle Identity Federation with Oracle Single Sign-On

This section describes the steps needed to install and deploy Oracle Identity Federation so that it is integrated with Oracle Single Sign-On.

Deployed in this manner, Oracle Identity Federation can leverage the authentication capabilities offered by Oracle Single Sign-On when local user authentication is required. Oracle Identity Federation can:

- integrate with OHS and Oracle Single Sign-On to serve as the authentication engine
- integrate with Oracle Single Sign-On to serve as the SP integration Module.

Briefly, the steps to achieve this deployment are:

- Create and Manage the Oracle HTTP Server Instance
- Integrate Oracle Single Sign-On with OHS
- Configure Oracle Identity Federation to use Oracle Single Sign-On as the Authentication Engine
- Configure Oracle Identity Federation for Oracle Single Sign-On SP Integration

- Configure Oracle Single Sign-On
- Testing Federated Single Sign-On

Detailed instructions for these steps follow.

Note: Oracle Identity Federation does not support the ability to force re-challenging the user for credentials when integrated with Oracle Single Sign-On, so that Oracle Identity Federation cannot support use cases where re-authentication must be forced.

For example, if an SP sends an `AuthnRequest` with `ForceAuthn="true"` to an Oracle Identity Federation IdP, and Oracle Identity Federation is integrated with Oracle Single Sign-On, the `ForceAuthn` flag is ignored.

3.2.2.1 Create and Manage the Oracle HTTP Server Instance

To configure OHS, follow the instructions in Section 3.2.1, "Deploying Oracle Identity Federation with Oracle HTTP Server".

3.2.2.2 Integrate Oracle Single Sign-On with OHS

This integration is necessary to deploy Oracle Identity Federation with Oracle Single Sign-On.

Register Partner Application

Start by registering the mod_osso module in 11*g* Release 1 (11.1.1) OHS with the 10*g* Oracle Single Sign-On server as a partner application.

For details on this procedure, refer to Configuring and Administering Partner Applications in the *Oracle Application Server Single Sign-On Administrator's Guide* for 10*g*.

You need to run `ssoreg` from the Oracle Single Sign-On server to generate an `osso.conf` file and manually copy it to the partner application (`AS_INST` of the Oracle Identity Federation instance).

Here is an example for registering a remote partner application on an Oracle Single Sign-On server:

```
$ORACLE_HOME/sso/bin/ssoreg.sh
-site_name oif.server.com:7499
-config_mod_osso TRUE
-mod_osso_url http://oif.server.com:7499
-remote_midtier
-config_file oif.server.com.osso.conf
```

Restart the `OC4J_SECURITY` instance of the Oracle Single Sign-On Server.

After you run this command, a file named `oif.server.com.osso.conf` is created in the directory where the command was invoked. Copy that file to `$AS_INST/config/OHS/$OHS_NAME/`.

Set Up mod_osso

The next step is to set up `mod_osso`.

Copy `$AS_INST/config/OHS/$OHS_NAME/disabled/mod_osso.conf` to `$AS_INST/config/OHS/$OHS_NAME/moduleconf`. All files in the `moduleconf` directory are read when OHS is started.

Open the `$AS_INST/config/OHS/$OHS_NAME/moduleconf/mod_osso.conf` file. Set the `OssoConfigFile` directive to reference the Oracle Single Sign-On configuration file that was created and then copied to the OHS `config` directory:

```
OssoConfigFile ${ORACLE_INSTANCE}/config/${COMPONENT_TYPE}/ ${COMPONENT_
NAME}/oif.server.com.osso.conf
```

Add the `/fed/user/authnosso` URL to be protected by Oracle SSO Server, through the `Location` element.

Then the `mod_osso.conf` example would look like this:

```
LoadModule osso_module ${ORACLE_HOME}/ohs/modules/mod_osso.so

<IfModule mod_osso.c>
    OssoIpCheck off
    OssoIdleTimeout off
    OssoConfigFile ${ORACLE_INSTANCE}/config/${COMPONENT_TYPE}/
    ${COMPONENT_NAME}/oif.server.com.osso.conf

    <Location /fed/user/authnosso>
       require valid-user
       AuthType Osso
    </Location>
</IfModule>
```

Note: If SSL is not enabled, you must add the `OssoSecureCookies` directive and set it to `off`; otherwise `mod_osso` will set the cookie as secure and instruct the browser to only send the cookie over HTTPS.

For details, see Secure Transmission of mod_osso Cookies in the *Oracle Enterprise Single Sign-On Suite Plus Administrator's Guide*.

The `mod_osso.conf` file should look like this:

```
LoadModule osso_module ${ORACLE_HOME}/ohs/modules/mod_osso.so

<IfModule mod_osso.c>
    OssoIpCheck off
    OssoIdleTimeout off
    OssoConfigFile ${ORACLE_INSTANCE}/config/${COMPONENT_TYPE}/${COMPONENT_
NAME}/oif.server.com.osso.conf

</IfModule>

#
#
# To have short hostnames redirected to fully qualified
# hostnames for clients that need authentication via
# mod_osso to be able to enter short hostnames into their
# browsers use a mod_rewrite configuration such as the following.
#
# e.g
#RewriteEngine On
#RewriteCond %{HTTP_HOST} !www.example.com
#RewriteRule ^.*$ http://%{SERVER_NAME}%{REQUEST_URI} [R]
#where www.example.com is the fully qualified domain name.
```

Restart OHS for the configuration changes on `mod_osso` and `mod_wl` to take effect.

3.2.2.3 Configure Oracle Identity Federation to use Oracle Single Sign-On as the Authentication Engine

In this task you configure the server to use the Oracle Single Sign-On authentication engine. For more information, see Section 2.3.1, "Engines in Oracle Identity Federation".

Start with these steps to enable the Oracle Single Sign-On authentication engine:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Authentication Engines**, then **Oracle SSO**.
3. Enable the engine.
4. Save the changes.

If Oracle Single Sign-On is only integrated as an authentication engine for Oracle Identity Federation, you will need to set up the Oracle Single Sign-On server to configure it for logout, and configure the Oracle Identity Federation server to configure it for logout.

To set up Oracle Single Sign-On:

1. Open `$ORACLE_HOME/sso/conf/policy.properties`.
2. Uncomment `SASSOLogoutUrl`.
3. Set the Oracle Identity Federation hostname/port information:

   ```
   SASSOLogoutUrl = http\://oif-hostname\:oif-port/fed/user/authnsloosso
   ```

4. Save the changes and exit.
5. Restart the `OC4J_SECURITY` instance.

> **See Also:** Section 5.15, "Configuring Authentication Engines"

To set up Oracle Identity Federation:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Authentication Engines**, then **Oracle SSO**.
3. Enable logout.
4. Enter the Oracle Single Sign-On **Server Logout URL**:

   ```
   http://osso-hostname:osso-port/sso/logout
   ```

5. Save the changes.

> **See Also:** Section 5.15, "Configuring Authentication Engines"

3.2.2.4 Configure Oracle Identity Federation for Oracle Single Sign-On SP Integration

This task involves enabling Oracle Single Sign-On as a service provider integration module, and if needed, disable logout for the Oracle Single Sign-On authentication engine.

First, take these steps to enable the Oracle Single Sign-On SP Module in Oracle Identity Federation:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Service Provider Integration Modules**, then **Oracle SSO**.
3. Enable the SP Module.
4. Select the authentication mechanism that will be used to locally authenticate users if federated identities are used during Federation SSO and if a federation record needs to be created during the SSO operation.
5. Enter the username attribute that Oracle Identity Federation needs to provide to Oracle SSO. Default is `uid`.
6. Enter the Oracle Single Sign-On server login URL:

   ```
   http://osso-hostname:osso-port/sso/auth
   ```

7. Enter the Oracle Single Sign-On server logout URL:

   ```
   http://osso-hostname:osso-port/sso/logout
   ```

8. Check **Logout Enabled**.
9. Click **Regenerate OSSO Secret** to create an encryption key that will be saved in a file and provided to Oracle Single Sign-On. Save the keystore locally.
10. Save the changes.

See Also: Section 5.16, "Configuring SP Integration Modules"

If Oracle Single Sign-On is integrated as an authentication engine for Oracle Identity Federation and an SP integration module, then the Oracle Single Sign-On authentication engine logout must be disabled, as the SP integration module is managing the logout. To disable the logout for Oracle Single Sign-On authentication engine in Oracle Identity Federation:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Authentication Engines**, then **Oracle SSO**.
3. Disable logout.
4. Save the changes.

See Also: Section 5.15, "Configuring Authentication Engines"

3.2.2.5 Configure Oracle Single Sign-On

This part of the setup requires setup on the Oracle Single Sign-On server, and partner configuration.

Note: A partner application is an Oracle Application Server-based application or a non-Oracle application that delegates the authentication function to the Oracle Single Sign-On server. A partner application is responsible for determining whether a user authenticated by Oracle Single Sign-On is authorized to use the application.

To set up Oracle Single Sign-On:

1. Copy the keystore file previously generated to the `$ORACLE_HOME/sso/conf` location and save it as a keystore.
2. Open `$ORACLE_HOME/sso/conf/policy.properties`.
3. Uncomment `SASSOAuthnUrl`, `SASSOLogoutUrl`, `SASSOAuthLevel` and `MediumHighSecurity_AuthPlugin`.
4. Set the Oracle Identity Federation hostname/port information for:
   `SASSOAuthnUrl= http\://oif-hostname\:oif-port/fed/user/sposso`
5. Set the Oracle Identity Federation hostname/port information for:
   `SASSOLogoutUrl = http\://oif-hostname\:oif-port/fed/user/spsloosso`
6. Set the authentication level for the Oracle Identity Federation plugin:
   `SASSOAuthLevel = MediumHighSecurity`
7. Set the `MediumHighSecurity_AuthPlugin`, which will define the Oracle Identity Federation plug-in: `MediumHighSecurity_AuthPlugin = oracle.security.sso.server.auth.SASSOAuth`
8. Save the changes and exit.
9. Restart the `OC4J_SECURITY` instance.

To configure a partner to use Oracle Identity Federation as the authentication plug-in:

1. Open `$ORACLE_HOME/sso/conf/policy.properties`.
2. Add the partner application to be protected by the authentication level mapped to the Oracle Identity Federation plug-in. For example:

   ```
   content.example.com\:8888 = MediumHighSecurity
   ```

3. Save the changes and exit.
4. Restart the `OC4J_SECURITY` instance.

3.2.2.6 Testing Federated Single Sign-On

Take these steps to test your federated single sign-on setup:

1. Use a web browser to access a protected resource. When prompted by the identity provider, log in using credentials in the IdP's domain. When prompted by the service provider, log in using credentials in the SP's domain. You should now be redirected to the protected resource.

   For details about protecting partner applications and resources see *Oracle Enterprise Single Sign-On Suite Plus Administrator's Guide*.

2. Log out, and then try to access the protected resource again. You should be prompted for login only by the identity provider.

3.2.3 Deploying Oracle Identity Federation with Oracle Access Manager 11g

You can integrate identity federation with the Oracle Access Access Manager server (OAM Server) 11*g* Release 1.

For integration details, see:

http://docs.oracle.com/cd/E23943_01/doc.1111/e15740/oif.htm

3.2.4 Deploying Oracle Identity Federation with Oracle Access Manager 10g

This section describes the steps needed to install and deploy Oracle Identity Federation so that it is integrated with Oracle Access Manager 10*g*. Integration enables Oracle Identity Federation to interact with Oracle Access Manager to create an authenticated user session. The steps illustrate a deployment scenario consisting of two nodes.

The section is broken out into separate instructions for the different component installation and deployment tasks:

- Create and Manage OHS
- Integrate Oracle Access Manager as an Authentication Engine
- Integrate Oracle Access Manager as an SP Integration Module

---

Note: Oracle Access Manager policy objects created by Oracle Identity Federation should not be changed from within Oracle Access Manager Policy/Access Manager administration interface; these objects can be identified by the description "Created by OIF. Do not modify". Modifying such objects from Oracle Access Manager can lead to synchronization issues; always update these policy objects from Fusion Middleware Control.

Note: Oracle Identity Federation does not support the ability to force re-challenging the user for credentials when integrated with the Oracle Access Manager 10*g* authentication engine, so that Oracle Identity Federation cannot support use cases where re-authentication must be forced.

For example, if an SP sends an `AuthnRequest` with `ForceAuthn="true"` to an Oracle Identity Federation IdP, and Oracle Identity Federation is integrated with Oracle Access Manager, the `ForceAuthn` flag is ignored.

3.2.4.1 Create and Manage OHS

The steps to create and manage your 11*g* Release 1 (11.1.1) OHS component instance are:

- Create the OHS instance.
- Link OHS instance to the Oracle WebLogic Server managed server where Oracle Identity Federation is running.
- Update the Oracle Identity Federation configuration.
- Manage the OHS instance.

See Also: Understanding Key Oracle Fusion Middleware Concepts in the *Oracle Fusion Middleware Administrator's Guide*.

Create the Oracle HTTP Server Instance

Note: `$OHS_NAME` refers to the name of the OHS component.

Execute this command to create the instance:

```
$AS_ISNT/bin/opmnctl createcomponent -componentType OHS -componentName $OHS_NAME
```

Link Oracle HTTP Server Instance to the Managed Server

To enable OHS to connect to the managed server where Oracle Identity Federation runs, take these steps:

1. Open `$AS_ISNT/config/OHS/$OHS_NAME/moduleconf/oif.conf`.
2. If the Idm installation is in standalone mode, uncomment and set the `WebLogicHost` and `WebLogicPort` variables to reference the WLS managed server where Oracle Identity Federation is running (for example `myhost.mycorp.com` and `7499`).
3. If the Idm installation is in clustered mode, uncomment and set the `WebLogicCluster` variable to reference the WLS managed servers where Oracle Identity Federation is running (for example `myhost1.mycorp.com:7499`, `myhost2.mycorp.com:7499`).
4. Save the changes, exit, and restart OHS.

See Also: "Section 3.2.1.2, "Manage the Oracle HTTP Server Instance" for restart instructions.

See Also: Understanding Key Oracle Fusion Middleware Concepts in the *Oracle Fusion Middleware Administrator's Guide*.

Update the OHS Configuration in Oracle Identity Federation

To enable the Oracle Identity Federation server to recognize OHS, take these steps:

1. Locate the Oracle Identity Federation instance in Fusion Middleware Control.
2. Navigate to **Administration**, then **Server Properties**.
3. Change the hostname to reflect the hostname configured in OHS, if different.
4. Change the Port/SSL Enabled and SOAP Port/SSL Enabled information to reflect the OHS configuration.
5. Save the changes.
6. Since the Oracle Identity Federation metadata will have changed, redistribute the metadata to any remote partners to notify them of the changes.

See Also: Section 5.2, "Configuring Server Properties"

Manage the Oracle HTTP Server Instance

To start OHS:

```
$AS_ISNT/bin/opmnctl startproc process-type=OHS
```

where `$AS_INST` is the application server instance home.

To stop OHS:

```
$AS_ISNT/bin/opmnctl stopproc process-type=OHS
```

To restart OHS:

```
$AS_ISNT/bin/opmnctl restartproc process-type=OHS
```

3.2.4.2 Integrate Oracle Access Manager as an Authentication Engine

> **See Also:** Section 2.3, "Authentication Engines" to review the features and benefits of authentication engines.

Oracle Access Manager as an authentication engine protects the `/fed/user/authnoam` URL.

The steps to configure this feature are:

- Verify requirements and update default OHS configuration
- Integrate OHS with Oracle Access Manager
- Configure Oracle Access Manager
- Configure Oracle Identity Federation

Verify Requirements

Take these steps to verify component versions:

1. Verify that the Oracle Access Manager server is at Version 10.1.4.3.
2. Verify that the Oracle Access Manager WebGate is installed for 11gR1 OHS (2.2).

Take these steps at the OHS:

1. Edit the `$AS_ISNT/config/OHS/$OHS_NAME/httpd.conf` file.
2. Uncomment the User and Group directives in the `httpd.conf` file, or add it.
3. Replace the default values of `nobody` with the user and users group that installed OHS 11*g* Release 1 (11.1.1), or add the user and users group if User and Group directives were missing.

   For example:

   ```
   User oracle
   Group dba
   ```

Integrate Oracle HTTP Server with Oracle Access Manager

Use the Oracle Access Manager console to create a new AccessGate and associate it with an Access Server instance.

> **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* 10*g* for details about the Web-based user interface.

> **See Also:** *Oracle Access Manager Installation Guide* 10*g* for details about installing WebGate.

On the machine where OHS is installed, install WebGate using these steps:

1. Launch the Install Wizard to install WebGate for Oracle HTTP Server11g
2. Enter the AccessGate details and the Access Server connection information.
3. Enter the location of the Oracle HTTP Server httpd.conf file (it should be `$AS_ISNT/config/OHS/$OHS_NAME/httpd.conf`) in order for WebGate to integrate with Oracle HTTP Server.

Configure Oracle Access Manager

Oracle Access Manager needs to protect an Oracle Identity Federation resource through WebGate. Since the WebGate is already installed, Oracle Access Manager needs only to have a policy that will protect the Oracle Identity Federation server in order to use Oracle Access Manager as an authentication engine. Once Oracle Access Manager authenticates, it will need to provide to Oracle Identity Federation the user identifier as an HTTP header.

Take these steps to configure Oracle Access Manager:

> **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* 10*g* for details about the Web-based user interface.

1. Go to the Oracle Access Manager console and navigate to the **Policy Manager**.
2. Protect the `/fed/user/authnoam` resource in a domain with an authentication scheme and an authorization rule.
3. In the **Authorization Rule**, go to the **Actions** tab, click **Modify** and in the **Authorization Success** section, add a Return property that will be an HTTP header with a value set to the user ID. (For example, set Type to `headervar`, set Name to `userid`, set Return Attribute to `uid`.)

> **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* 10*g*

Configure Oracle Identity Federation

Take these steps:

1. Locate the Oracle Identity Federation instance in Fusion Middleware Control.
2. Navigate to **Administration**, then **Authentication Engines**, then **Oracle Access Manager**.
3. Enable the Engine.
4. Enter the HTTP header containing the User ID.
5. Perform these logout configuration steps if Oracle Identity Federation and Oracle Access Manager are integrated only through the authentication engine framework. Otherwise, disable logout for Oracle Access Manager authentication Engine here and see Section 3.2.4.3, "Integrate Oracle Access Manager as an SP Integration Module" to configure logout operation.
   - Check the **Logout Enabled** box if logout needs to be enabled.
   - Check the **Clear Cookie** box if resetting the Oracle Access Manager cookie is enough for Oracle Identity Federation to log the user out of the Oracle Access Manager domain.
   - Check **Redirect to Logout URL** and fill in the URL if Oracle Identity Federation needs to redirect the user to a specific URL for Oracle Access Manager logout. See the note below for more information.
6. Save the changes.

Note: When the user needs to be redirected to an Oracle Access Manager URL for logout (in case Oracle Access Manager needs to perform extra operations), you need to configure Oracle Identity Federation by checking the **Redirect to Logout URL** box, and entering the URL to which the user is redirected. When Oracle Identity Federation redirects the user to that URL, it will append a return URL as a query parameter; this is the Oracle Identity Federation URL to which the user is redirected after performing the extra Oracle Access Manager operations.

The query parameter to be appended to the Oracle Access Manager logout URL is referenced by `returnurl`.

Note: The fix for Oracle Access Manager bug 5736326 is required when protecting the `/fed/user/authnoam` URL with HTTP Basic Authentication.

Using an Alternate Return Attribute for the HTTP Header

When Oracle Identity Federation is integrated with Oracle Access Manager for authentication, WebGate is protecting the `/fed/user/authnoam` URL, and Oracle Access Manager is configured to pass the user identifier as an HTTP header to Oracle Identity Federation, the policy protecting the `/fed/user/authnoam` URL contains an authorization rule with an action that adds an HTTP header with a return attribute referencing the user ID from the LDAP user record. This return attribute is the same as the Unique User ID set in Fusion Middleware Control when you navigate to the Oracle Identity Federation instance, under **Administration**, then **Data Stores**, then **User Data Store** section.

Due to a bug, `orclguid` cannot be used as the return attribute for the HTTP header containing the user identifier. As a workaround, the unique user identifier must be changed to another attribute. To perform the change:

- change the return attribute in the Oracle Access Manager console to the new attribute (uid for example).
- in Fusion Middleware Control navigate to **Oracle Identity Federation Administration**, then **Data Stores**, then **User Data Store**, and change the Unique User ID to the new attribute (uid for example).
- if other authentication engines were used, check that their Unique User ID attributes is correctly updated.
- if Oracle Identity Federation was integrated with Oracle Access Manager through the Oracle Access Manager SP Integration Module, update the integration: after performing the above changes, navigate to the Oracle Identity Federation instance in Fusion Middleware Control, then **Administration**, then **SP Integration Modules**, then **OAM SP Engine**, enter the Oracle Access Manager administrator credentials, select the created authentication schemes to be updated, and click "**Configure Oracle Access Manager**"; this updates the mapping rules in Oracle Access Manager to reflect the new attribute.

3.2.4.3 Integrate Oracle Access Manager as an SP Integration Module

This task enables the SP integration module to interact with Oracle Access Manager.

The basic steps are:

- Verify requirements
- Install Oracle Access Server SDK
- Integrate Oracle Access Manager with Oracle Access Server SDK
- Update the Oracle WebLogic Server Classpath
- Configure Oracle Identity Federation
- Integrate Oracle Identity Federation with Oracle Access Manager
- Protect an Oracle Access Manager Resource with Oracle Identity Federation

Verify Requirements

Take these steps:

1. Ensure Access Server SDK 10*g* is installed.
2. In a high availability (HA) environment, the Access Server SDK needs to be installed on different machines, and integrated as different AccessGates with Oracle Access Manager.

> **Note:** When deploying in an HA environment, be sure to read and complete the instructions in the section High Availability Considerations for Integration with Oracle Access Manager in the *Oracle Fusion Middleware High Availability Guide*.
>
> Be sure to follow the directions regarding the directory where the Access Server SDK is installed, and restart all managed servers.

Install Oracle Access Server SDK

Configure Oracle Identity Federation to reference the directory where the SDK is installed.

If the SDK is installed in the Domain Home directory, you can reference the SDK folder relative to the Domain Home folder; otherwise, Oracle Identity Federation needs to reference the SDK folder using an absolute path.

To use Oracle Identity Federation in an HA environment, it is preferable to install the Access Server SDK under the Domain Home folder, using the same directory name relative path on the different machines where Oracle Identity Federation is installed. This way, the different Oracle Identity Federation instances share the same configuration; specifically, the directory where the Access Server SDK is installed has the same value for all the Oracle Identity Federation instances.

Integrate Oracle Access Manager with Oracle Access Server SDK

This task enables a new AccessGate to be associated with an Access Server instance.

On the Oracle Access Manager console, create the new AccessGate with Access Management Service enabled, and associate it with the Access Server instance.

> **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* 10*g* for details about the Web-based user interface.

Integrate the Access Server SDK by invoking the `configureAccessGate` script:

```
$ACCESS_SERVER_SDK/oblix/tools/configureAccessGate -i $ACCESS_SERVER_SDK -t
AccessGate -w $ACCESS_GATE_ID -m open -h $ACCESS_SERVER_HOST -p $ACCESS_SERVER_
```

```
PORT
```

replacing:

- `$ACCESS_SERVER_SDK` by the absolute path of the Access Server SDK directory
- `$ACCESS_GATE_ID` by the identifier for this Access Gate
- `$ACCESS_SERVER_HOST` by the hostname of machine where the Access Server is installed
- `$ACCESS_SERVER_PORT` by the port of the Access Server.

Update the Oracle WebLogic Server Environment

The managed server where Oracle Identity Federation is running needs to be able to access the JAR file and the shared libraries required for Oracle Access Manager integration.

> **See Also:** Understanding Key Oracle Fusion Middleware Concepts in the *Oracle Fusion Middleware Administrator's Guide*.

To update the environment:

1. Stop managed server.

   > **See Also:** Starting and Stopping WebLogic Servers in the *Oracle Fusion Middleware Administrator's Guide*.

2. Copy the `jobaccess.jar` file from the `$ASDK_DIR/oblix/lib` folder to the `$DOMAIN/lib` folder.
3. The next step depends on the version of Oracle Access Manager you are running.
   - If you are at Oracle Access Manager version 10.1.4.3 or later and want to start the managed server from the WebLogic administration console, follow these steps:
     - Open the WebLogic administration console.
     - Navigate to **Servers**, then **Managed Server**, then **Configuration**, then **Server Start**.
     - In the text box for **Arguments**, append `-Djava.library.path` to include `ASDK_DIR/oblix/lib`.
     - Save the changes.
     - Start the managed server from the console.

     > **See Also:** Understanding Key Oracle Fusion Middleware Concepts in the *Oracle Fusion Middleware Administrator's Guide*.

- If you are at Oracle Access Manager version 10.1.4.2 or earlier, or if you are at Oracle Access Manager version 10.1.4.3 and wish to start managed server from the command line:
  - Copy the `jobaccess.jar` file from the `$ASDK_DIR/oblix/lib` folder to the `$DOMAIN/lib` folder.
  - Stop managed server.
  - On Linux, open the file `$DOMAIN/bin/startManagedWebLogic.sh`:

Add the following lines:

```
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:accessSDK installdir/oblix/lib
export LD_LIBRARY_PATH
```

If the Oracle Access Manager version is 10.1.4.2 or earlier, add the following two lines as well:

```
LD_ASSUME_KERNEL=2.4.19
export LD_ASSUME_KERNEL
```

– On Windows, open the file `$DOMAIN/bin/startManagedWebLogic.cmd`:

  Add the following line:

  ```
  set PATH=%PATH%;AccessSDK_InstallDir/oblix/lib
  ```

– Start managed server from the command line (not the administration console).

Configure Oracle Identity Federation

Take these steps:

1. Locate the Oracle Identity Federation instance in Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider Integration Modules**, then **Oracle Access Manager**.
3. Enable the SP module.
4. Select the authentication mechanism to use when processing a SAML assertion (required in some cases when a local federation record needs to be created for the user).
5. Enter the directory where the Access Server SDK is installed. If the SDK is installed under the `$DOMAIN_HOME` directory, then the path to the folder can be relative to `$DOMAIN_HOME`; otherwise the path must be absolute.
6. Enter the default authentication scheme that Oracle Identity Federation should use when creating user sessions in Oracle Identity Federation.
7. Enter the cookie domain that Oracle Identity Federation will set when creating the Oracle Access Manager cookie.
8. Oracle Identity Federation can set the Oracle Access Manager cookie as either a persistent cookie or a session cookie. For a persistent cookie, enter the time in minutes during which the cookie will be valid; for a session cookie, enter 0.
9. Check whether the cookie should be marked as secure: in this case, the browser will send the cookie over an HTTPS connection.
10. Check the **Logout Enabled** box if logout needs to be enabled (recommended).
11. Check the **Clear Cookie** box if resetting the Oracle Access Manager cookie is enough for Oracle Identity Federation to log the user out of the Oracle Access Manager domain.
12. Check **Redirect to Logout URL** and fill in the URL if Oracle Identity Federation needs to redirect the user to a specific URL for Oracle Access Manager logout.

Note: When the user needs to be redirected to an Oracle Access Manager URL for logout (in case Oracle Access Manager needs to perform extra operations), you need to configure Oracle Identity Federation by checking the **Redirect to Logout URL** box, and entering the URL to which the user is redirected. When Oracle Identity Federation redirects the user to that URL, it will append a return URL as a query parameter; this is the Oracle Identity Federation URL to which the user is redirected after performing the extra Oracle Access Manager operations.

The query parameter to be appended to the Oracle Access Manager logout URL is referenced by `returnurl`.

13. Save your changes.

Note:

- The cookie domain must be set on the Webgate for the protected resource. An example of a cookie domain is:

  ```
  .us.example.com
  ```

- You use Fusion Middleware Control to configure the user data store that Oracle Identity Federation uses when creating policy objects in the Oracle Access Manager Policy Server.

  If you change the user data store through Fusion Middleware Control:

  - redo the Oracle Identity Federation/Oracle Access Manager integration
  - update the existing authentication schemes that were created by Oracle Identity Federation in the Oracle Access Manager Policy Server.

Integrate Oracle Identity Federation with Oracle Access Manager

After processing an incoming SSO assertion and identifying the user, Oracle Identity Federation will create an Oracle Access Manager session for that user in the Oracle Access Manager domain. To do so, Oracle Identity Federation will:

1. Use a policy domain created by Oracle Identity Federation at configuration time.
2. Map the Oracle Identity Federation authentication mechanism, representing the authentication method used by the IdP to challenge the user, to an Oracle Access Manager authentication scheme that was created by Oracle Identity Federation at configuration time. If the mapped Oracle Access Manager authentication scheme does not exist, then Oracle Identity Federation will use the default authentication scheme entered in the Oracle Identity Federation configuration section
3. Interact with Oracle Access Manager to create the user session, by specifying the policy domain and the authentication scheme for that session.

   The policy domain name that you enter for Oracle Identity Federation must reference an existing policy domain that was created by Oracle Identity

Federation.

4. Set the Oracle Access Manager cookie in the user's browser

For proper integration, Oracle Identity Federation needs to create policy objects and authentication schemes in Oracle Access Manager. Perform the following operations:

Note: This task assumes you have the appropriate administrator credentials for Oracle Access Manager. Ensure that the Oracle Access Manager **Master Administrators** account is used to create the policy objects.

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* 10*g*

See Also: Section 5.16.3, "SP Integration Module - Oracle Access Manager 10g" for screen details.

1. Locate the Oracle Identity Federation instance in Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider Integration Modules**, then **Oracle Access Manager**.
3. Expand the **Oracle Access Manager Properties** section.
4. Enter the Oracle Access Manager credentials to configure Oracle Access Manager.

   Note: Credentials will only be used to connect to the Oracle Access Manager Server for configuration when you click the **Configure Oracle Access Manager** button; these credentials are not stored in any Oracle Identity Federation configuration file.

5. Enter the Host ID that Oracle Identity Federation must use when configuring the policy domain. Define the Host ID in the Oracle Access Manager server. The Host ID must contain the hostname:port information that the Oracle Identity Federation server is configured to use, and its variations.

   Note: This is a required parameter.

6. Enter the default authorization rule that will be used when creating the policy domain.
7. The available Oracle Identity Federation authentication mechanisms are listed in the table; for each, the table lists the mapped authentication scheme name and its authentication scheme level. These mappings are stored in Oracle Identity Federation only by default, and you need to select the mechanisms and schemes to be created, updated, or deleted in Oracle Access Manager.

   When you select a scheme to create, and click Configure Oracle Access Manager, the scheme is created with the specified name and level, and mapped to the corresponding authentication mechanism in the Oracle Identity Federation configuration. You must select one of the created schemes as the default Oracle Access Manager authentication scheme used by Oracle Identity Federation. By default this value is password-protected by Oracle Identity Federation, so if nothing is selected as default, the password-protected authentication scheme must be selected for create. If you select a scheme to delete, likewise, the scheme is deleted from Oracle Access Manager. If you select a scheme for update, the default Oracle Access Manager authentication scheme used by Oracle Identity Federation, and its name, are updated in Oracle Access Manager.

Note: In order that the server can update/delete the authentication scheme, it must not be in use by any domains.

8. Click **Configure Oracle Access Manager**.

Protect an Oracle Access Manager Resource with Oracle Identity Federation

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* for 10*g*.

After integrating Oracle Identity Federation with Oracle Access Manager and creating authentication schemes, you can now protect resources using the schemes you have created. Protecting a resource with a specific scheme has the following effect:

1. When a non-authenticated user (or an authenticated user with authentication level lower than that of the scheme) tries to access a resource protected by an Oracle Identity Federation authentication scheme, the Oracle Access Manager server redirects the user to Oracle Identity Federation for Federation SSO.
2. Oracle Access Manager provides Oracle Identity Federation the resource being requested and the Oracle Identity Federation authentication scheme name to be used.
3. Oracle Identity Federation maps that authentication scheme to an authentication mechanism, and then to a SAML/WS-Fed authentication method.
4. Oracle Identity Federation starts the Federation SSO flow by sending the user to an identity provider and by specifying the authentication method to use in challenging the user for authentication.
5. The IdP will challenge the user, create an assertion and send the user back to Oracle Identity Federation with the assertion.
6. Oracle Identity Federation processes the assertion, extracting from it the method used to authenticate the user and map it to an authentication mechanism.
7. After successful processing, Oracle Identity Federation maps the authentication mechanism to an authentication scheme and creates an Oracle Access Manager session for the user.
8. Oracle Identity Federation redirects the user to the requested resource.
9. Finally Oracle Access Manager grants access to the resource for the authenticated user.

3.2.5 Oracle Identity Federation/SP Authenticating to Oracle Access Manager

You can configure Oracle Identity Federation, when acting as service provider, to authenticate itself to the Oracle Access Manager server when creating an Oracle Access Manager user session.

Topics in this section include:

3.2.5.1 Authentication Overview

This authentication operation occurs when Oracle Identity Federation uses an Oracle Access Manager federation authentication scheme, through the AccessGate installed on the machine hosting Oracle Identity Federation, to create a user session.

The operation ensures that the module invoking the scheme is indeed the Oracle Identity Federation server and that no other process is trying to use the scheme.

Operational Flow

The deployment and run-time flow are as follows:

1. Using Fusion Middleware Control, the Oracle Identity Federation administrator creates/updates the existing Oracle Access Manager federation schemes to add two new plugins
2. Using Fusion Middleware Control, the Oracle Identity Federation administrator provides the necessary credentials to OIF
3. At runtime, Oracle Identity Federation passes the credentials along with the data used to create the Oracle Access Manager user session
4. The Oracle Access Manager server validates the Oracle Identity Federation credentials against the LDAP user repository
5. After the Oracle Identity Federation credentials are validated, an Oracle Access Manager user session is created

Customizing the LDAP Account

The administrator can customize the LDAP account used to validate the Oracle Identity Federation credentials to select:

- the location of the entry (that is, location different from the user's branch)
- the object class of the entry

Servers Authorized to Invoke Authentication

For security reasons, the Oracle Identity Federation username can be set in the credential_mapping plugin of the federation scheme. This ensures that only the user corresponding to that account can be used when invoking this scheme.

This feature is optional, but enabling it ensures that only authorized Oracle Identity Federation servers invoke the federation authentication schemes.

3.2.5.2 Enabling Authentication with Existing Federation Schemes

In this scenario:

- Oracle Identity Federation is already deployed and integrated with Oracle Access Manager
- Oracle Identity Federation is not configured for authentication to Oracle Access Manager

- no federation schemes created in the Access server are configured for Oracle Identity Federation authentication to Oracle Access Manager

The configuration involves:

- creating an account in the LDAP directory to use for Oracle Identity Federation authentication
- setting Oracle Identity Federation account information to enable authentication using Fusion Middleware Control
- updating all existing federation schemes to define the new Oracle Access Manager plugins using Fusion Middleware Control
- updating all existing federation schemes to define new authentication flows for the new plugins using the Oracle Access Manager console

Create the LDAP Entry

So that Oracle Identity Federation can authenticate to Oracle Access Manager when using a federation scheme, the LDAP directory must contain an entry to use in validating the Oracle Identity Federation credentials. If no such entry exists, create one that is both searchable based on an identifier and has a password attribute. You use Fusion Middleware Control to set:

- the identifier
- passwords
- the base DN of the entry
- the entry's object class
- the attribute to contain the identifier

You can choose:

- a location for the entry different from the branch where all user records are located
- an object class different from the user record type

Set Up Oracle Identity Federation Account Information

To configure Oracle Identity Federation to present credentials when invoking a federation scheme, take these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **SP Integration Modules**, then **Oracle Access Manager**.
3. Check the box to enable Oracle Identity Federation authentication.
4. Enter the username and password of the account to use for Oracle Identity Federation authentication.
5. Enter the Base DN referencing the location where the Oracle Identity Federation account is located.
6. Enter the object class of the LDAP entry to use for Oracle Identity Federation authentication.
7. Enter the LDAP entry attribute that will contain the username and is searchable (for example, `uid` if it is defined in the LDAP entry).

Define New Plug-ins

Next update the existing federation schemes to include two new plugins used for the authentication operation. Take these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **SP Integration Modules**, then **Oracle Access Manager**.
3. Select and click **Update** for all the federation schemes that were created in Oracle Access Manager.
4. Enter the Oracle Access Manager administrator credentials to enable administrative update operations to be performed on the Oracle Access Manager server.
5. Click **Configure Oracle Access Manager**.

This updates the schemes with two new plugins:

- a credential_mapping plugin used to locate the Oracle Identity Federation account.
- a validate_password plugin used to validate the password provided by Oracle Identity Federation against the one from the Oracle Identity Federation account.

Define New Authentication Flows

Due to Access Server limitations, you cannot use Fusion Middleware Control to create new authentication flows to uptake the new plugins created in the previous step; instead, you must use the Oracle Access Manager console to create those objects.

For each federation scheme you updated, perform these actions:

1. Log in to the Oracle Access Manager console as administrator.
2. Navigate to the Access System Console, then Access System Configuration, then Authentication Management.
3. Select the federation scheme you wish to update.
4. Click the Steps tab.
5. Click **Add** to add a new step.
6. Enter a name for the step in the "Step Name" field.
7. In the "Available Plugins" table, perform these steps in order:

   Note: The order is important.

   a. Select the second credential_mapping plugin and click **add**.
   b. Select the validate_password plugin and click **add**.
   c. Select the first credential_mapping plugin and click **add**.
8. Click **Save**.
9. Click the Authentication Flow tab.
10. Click **Modify**.
11. Select the new step as the "Initiating Step".
12. Select Stop for "On Success Next Step".

13. Select Stop for "On Failure Next Step".
14. Click **Save**.
15. Click the Steps tab.
16. Select the old step. Click **Delete**.

3.2.5.3 Enabling Authentication when Creating New Federation Schemes

This configuration supports the following scenario:

- Oracle Identity Federation is already deployed and integrated with Oracle Access Manager, or Oracle Identity Federation is deployed but not yet integrated with Oracle Access Manager.
- Oracle Identity Federation is not configured for authentication to Oracle Access Manager.
- No federation schemes have been created in Access server.

The tasks include:

- creating the account in the LDAP directory used for Oracle Identity Federation authentication
- setting information about the Oracle Identity Federation account, and any Oracle Identity Federation/Oracle Access Manager integration which might involve creating new authentication schemes, in Fusion Middleware Control.
- creating new federation schemes using Fusion Middleware Control

Create the LDAP Account

So that Oracle Identity Federation can authenticate to Oracle Access Manager when using a federation scheme, the LDAP directory must contain an entry to use in validating the Oracle Identity Federation credentials. If no such entry exists, you must create one that is both searchable based on an identifier and has a password attribute. You use Fusion Middleware Control to set:

- the identifier
- passwords
- the base DN of the entry
- the entry's object class
- the attribute to contain the identifier

You can choose:

- a location for the entry different from the branch where all user records are located
- an object class different from the user record type

Set Up Oracle Identity Federation Account Information

To configure Oracle Identity Federation to present credentials when invoking a federation scheme, take these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **SP Integration Modules**, then **Oracle Access Manager**.
3. Enable Oracle Identity Federation authentication.

4. Enter the username and password of the account to use for Oracle Identity Federation authentication.
5. Enter the Base DN referencing the location where the Oracle Identity Federation account is located.
6. Enter the object class of the LDAP entry to use for Oracle Identity Federation authentication.
7. Enter the LDAP entry attribute that will contain the username and is searchable (for example, `uid` if it is defined in the LDAP entry).
8. Configure the rest of the fields for Oracle Access Manager as needed, and click **Configure Oracle Access Manager** to apply the changes.

Any federation schemes created subsequently will contain the necessary information to authenticate Oracle Identity Federation at runtime. No manual steps are needed in the Oracle Access Manager console to update the federation authentication schemes.

3.2.5.4 Updating Oracle Identity Federation Credentials

In this scenario:

- Oracle Identity Federation is already deployed and integrated with Oracle Access Manager.
- Oracle Identity Federation is configured for authentication to Oracle Access Manager.
- Federation schemes exist and are configured to perform Oracle Identity Federation authentication at run-time.

Updating Oracle Identity Federation credentials may necessitate updates to the federation schemes:

- if only the Oracle Identity Federation password is updated, federation schemes can remain unchanged
- if information about the Oracle Identity Federation account other than password (object class, base DN, username, and so on) is updated, all federation schemes must be updated.

Update Oracle Identity Federation Password Only

If updating only the Oracle Identity Federation password, take these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **SP Integration Modules**, then **Oracle Access Manager**.
3. Enter the password of the account to be used for Oracle Identity Federation authentication.
4. Click **Configure Oracle Access Manager** to apply the changes.

Update Other Data Besides Password

When updating other information besides the Oracle Identity Federation password, perform the following steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to SP Integration Modules, then Oracle Access Manager.

3. Update the username and/or password of the account to use for Oracle Identity Federation authentication, as needed.
4. Update the Base DN referencing the location where the Oracle Identity Federation account is located, as needed.
5. Update the object class of the LDAP entry to use for Oracle Identity Federation authentication, as needed.
6. Update the LDAP entry attribute that will contain the username and is searchable (for example, `uid` if it is defined in the LDAP entry), as needed.
7. Select and click **Update** for all the federation schemes that were created in Oracle Access Manager.
8. Enter the Oracle Access Manager administrator credentials to enable administrative update operations to be performed on the Oracle Access Manager server.
9. Click **Configure Oracle Access Manager**.

The federation schemes now contain updated information in the credential_mapping plugin for the Oracle Identity Federation authentication operation.

3.2.5.5 Disabling Authentication to Oracle Access Manager

In this scenario:

- Oracle Identity Federation is already deployed and integrated with Oracle Access Manager.
- Oracle Identity Federation is configured for authentication to Oracle Access Manager.
- Federation schemes exist and are configured to perform Oracle Identity Federation authentication at run-time.

The tasks involve:

- updating the existing federation schemes to remove objects involved in Oracle Identity Federation authentication
- disabling Oracle Identity Federation authentication with Fusion Middleware Control

Update Existing Federation Schemes

This procedure updates the federation schemes so that they no longer require Oracle Identity Federation authentication. If this is not done, the schemes would continue to expect credentials but Oracle Identity Federation would no longer be sending them.

For each of the federation scheme that was updated, take these steps:

1. Navigate to Access System Console, then Access System Configuration, then Authentication Management.
2. Select the federation scheme to update.
3. Select the **Steps** tab.
4. Click **Add** to add a new step.
5. Enter a name for the step in the "Step Name" field.
6. The next step depends on when Oracle Identity Federation authentication was enabled (before or after creation of this scheme).

- If the order of the plugins is credential_mapping, then validate_password, then credential_mapping, select the last credential_mapping plugin and click **add**.
- if the order of the plugins is credential_mapping, then credential_mapping, then validate_password, select the first credential_mapping plugin and click **add**.

7. Click **Save**.
8. Click the Authentication Flow tab.
9. Click **Modify**.
10. Select the new step as the "Initiating Step"
11. Select Stop for "On Success Next Step".
12. Select Stop for "On Failure Next Step".
13. Click **Save**.
14. Click the Steps tab.
15. Select the old step. Click **Delete**.
16. Click the Plugins tab.
17. Click **Modify**.
18. Select the validate_password plugin and the credential_mapping plugin with a plug-in parameter where the Oracle Identity Federation username is specified.
19. Click **Delete**, then **Save**.

After this action, the only remaining plugin is the credential_mapping plugin with a plugin parameter containing the string `%OIFUSERID%`.

Disable Oracle Identity Federation Authentication

Take these steps to disable authentication:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to SP Integration Modules, then Oracle Access Manager.
3. Disable Oracle Identity Federation authentication.
4. Click **Configure Oracle Access Manager** to apply the change.

After this action, Oracle Identity Federation no longer sends credentials for authentication when invoking a federation scheme.

To Re-enable Oracle Identity Federation Authentication

To re-enable the Oracle Identity Federation authentication to Oracle Access Manager after disabling it, follow the instructions in Section 3.2.5.2, "Enabling Authentication with Existing Federation Schemes".

3.2.6 Deploying Oracle Identity Federation with Oracle Directory Server Enterprise Edition

This section describes how to integrate Oracle Directory Server Enterprise Edition (formerly Sun Java System Web Server) with Oracle Identity Federation to serve as a web proxy.

This section contains the following topics:

- Requirements
- Configuring Oracle Identity Federation Without a Web Proxy Server
- Configuring Oracle Identity Federation Behind a Web Proxy Server
- Updating the Identity and Access Management servers
- Oracle Directory Server Enterprise Edition Sample Configuration Files

3.2.6.1 Requirements

When using a proxy in front of Oracle Identity Federation, the host name and port number of the proxy server instance are used to access Oracle Identity Federation, and are set in the Oracle Identity Federation configuration.

3.2.6.2 Configuring Oracle Identity Federation Without a Web Proxy Server

Start by installing Oracle Identity Federation, and configure and integrate it with any back-ends (LDAP, RDBMS, Oracle Access Manager, and others) required by your deployment. This includes:

- configuring Oracle Identity Federation for SSO SAML protocols
- adding trusted providers to Oracle Identity Federation's Federations
- integrating Oracle Identity Federation with back-ends

See Also:

- Chapter 4, "Server Administration"
- Chapter 5, "Configuring Oracle Identity Federation"

3.2.6.3 Configuring Oracle Identity Federation Behind a Web Proxy Server

When configuring Oracle Identity Federation behind a Web Proxy server, the steps are similar to the ones performed in a non-Web Proxy server environment as described in Section 3.2.6.2, "Configuring Oracle Identity Federation Without a Web Proxy Server", except that:

- The Web Proxy server will be configured to reference Oracle Identity Federation
- The Oracle Identity Federation configuration will use the hostname and port information of the Web Proxy Server

Follow the standard procedures to configure Oracle Identity Federation as shown in Section 3.2.6.2, "Configuring Oracle Identity Federation Without a Web Proxy Server", with these modifications:

- Change the configuration URLs to their respective proxy server URLs.

  In Fusion Middleware Control, navigate to Administration, then Server Properties.

- Collect the metadata using the proxy URLs, not actual URLs, then upload the metadata.

  Navigate to **Administration**, then **Security and Trust**, and retrieve the metadata (the hostname/port information in the metadata will now use the new values). Distribute the metadata to the remote providers.

- At the Access System console, create a host identifier in the format:

  ```
  proxy-host:port
  ```

and change the challenge redirect of the authentication scheme to *proxy-host:port*.

> **See Also:** Section 5.2, "Configuring Server Properties" and Section 5.10, "Configuring Security and Trust"

> **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* 10*g* for details about the Web-based user interface.

For details about configuring the Sun One proxy server for Oracle Identity Federation, see Guidelines for Modifying the obj.conf File in *Oracle Fusion Middleware Using Web Server Plug-Ins with Oracle WebLogic Server*.

3.2.6.4 Updating the Identity and Access Management servers

Users will now access Oracle Identity Federation through the Web server proxy, and the IAM servers like Oracle Access Manager also need to be updated so that they reference the proxy instead of the local Oracle Identity Federation machine.

Go to any back end that references Oracle Identity Federation (such as Oracle Access Manager or Oracle Single Sign-On), and update their configuration to use the hostname/port values of the Web proxy server instead of the local machine where Oracle Identity Federation is installed.

3.2.6.5 Oracle Directory Server Enterprise Edition Sample Configuration Files

The Web proxy server will need to forward the HTTP requests to the machine where Oracle Identity Federation is installed.

This section provides samples of the `obj.conf` and `magnus.conf` configuration files.

Sample obj.conf File

```
<Object name="default">
AuthTrans fn="match-browser" browser="*MSIE*" ssl-unclean-shutdown="true"
NameTrans fn="assign-name" from="/*" name="serverexample"
NameTrans fn="ntrans-j2ee" name="j2ee"
NameTrans fn=pfx2dir from=/mc-icons dir="/home/pfx/SunOne6.1/ns-icons"
name="es-internal"
NameTrans fn=document-root root="$docroot"
PathCheck fn=unix-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html,index.jsp"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD|POST) type=*~magnus-internal/* fn=send-file
Service method=TRACE fn=service-trace
Error fn="error-j2ee"
AddLog fn=flex-log name="access"
</Object>

<Object name="j2ee">
Service fn="service-j2ee" method="*"
</Object>
```

```
<Object name="cgi">
ObjectType fn=force-type type=magnus-internal/cgi
Service fn=send-cgi user="$user" group="$group" chroot="$chroot" dir="$dir"
   nice="$nice"
</Object>

<Object name="es-internal">
PathCheck fn="check-acl" acl="es-internal"
</Object>

<Object name="send-compressed">
PathCheck fn="find-compressed"
</Object>

<Object name="compress-on-demand">
Output fn="insert-filter" filter="http-compression"
</Object>


# Execute these instructions for any resource with the assigned name
# "server.example.com"
<Object name="serverexample">
# Proxy the requested resource to the URL
# "http://server.example.com:8080"
Service fn="service-passthrough" servers="http://unit1.mycorp.co.in:1234"
</Object>
```

Sample magnus.conf File

```
#
# The NetsiteRoot, ServerName, and ServerID directives are DEPRECATED.
# They will not be supported in future releases of the Web Server.
NetsiteRoot /home/pfx/SunOne6.1
ServerName calgary
ServerID https-oif_idp_flagstaff
#
RqThrottle 128
DNS off
Security off
PidLog /home/pfx/SunOne6.1/https-oif_idp_flagstaff/logs/pid
User pfx
StackSize 131072
TempDir /tmp/https-oif_idp_flagstaff-65cd125c

Init fn=flex-init access="$accesslog" format.access="%Ses->client.ip%
   - %Req->vars.auth-user% [%SYSDATE%] \"%Req->reqpb.clf-request%\"
   %Req->srvhdrs.clf-status% %Req->srvhdrs.content-length%"
Init fn="load-modules" shlib="/home/pfx/SunOne6.1/bin/https/lib/libj2eeplugin.so"
shlib_flags="(global|now)"
Init fn="load-modules"
shlib="/home/pfx/SunOne6.1/bin/https/passthrough/plugins/passthrough
   /libpassthrough.so"
```

3.2.7 Using the Test SP Engine

Oracle Identity Federation provides a test SP engine for the purpose of Single Sign-On testing. The following sections describe how to use the test SP engine.:

- Configure the Test SP Engine
- Use the Test SP Engine for SP-Initiated SSO
- Use the Test SP Engine with IdP-Initiated SSO
- Test SP Engine Results

Note: The test SP engine must be disabled in a production environment.

3.2.7.1 Configure the Test SP Engine

Take these steps to enable/disable the test SP engine:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Service Provider Integration Modules**.
3. In the Test SP tab, select/unselect **Enable Engine**.

To make the test SP engine the default SP engine, follow these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Service Provider Integration Modules**.
3. Select **Default SP Integration Module** to be Test SP (Note: The test SP engine must be enabled).

See Also: Section 5.15, "Configuring Authentication Engines"

3.2.7.2 Use the Test SP Engine for SP-Initiated SSO

To use the test engine, you can initiate a single sign-on flow at the following URL:

```
HTTP://OIF-SP-HOST:OIF-SP-PORT/fed/user/testspsso
```

where:

- `HTTP` is `http` for an open connection and https for a secure connection
- `OIF-SP-HOST` is the hostname of the Oracle Identity Federation service provider server
- `OIF-SP-PORT` is the `http` or `https` port number of the Oracle Identity Federation service provider server. Omit the entry for port 80 or https port 443.

When you click the **Start SSO** button, a request is sent to the Oracle Identity Federation service provider to start Single Sign-On (that is, an authentication request is sent to the identity provider) with the information provided on the page. You can specify the following parameters:

- IdP Provider ID: This is the Provider ID or description of the identity provider to which to the service provider will send the authentication request.
- Authn Request Binding: This specifies the binding the service provider will use to send the authentication request. Applies only to SAML 2.0 protocol
- Force Authentication: If checked, the identity provider is forced to authenticate the user, instead of possibly relying on a previous authentication context. Applies only to SAML 2.0/WS-Fed protocols.

- Is Passive: If checked, the identity provider must not interact with the user. Applies only to SAML 2.0 protocol.
- Relay State: In this field, you can enter any string. This is either an identifier for the request or a return URL and is returned as a relay state after the Oracle Sign-On flow has been performed.
- Use Default Configuration: If checked, Oracle Identity Federation will use the default configuration for the following properties:
  - Allow Federation Creation
  - SSO Response Binding
  - Name ID Format
  - Requested Authentication Mechanism
  - Authentication Mechanism Comparison

  If not checked, the following applies:
  - Allow Federation Creation: If not checked, the identity provider must not create a federation for the user, if one does not exist. Applies only to SAML 2.0 protocol
  - SSO Response Binding: This specifies the binding that the service provider will request the identity provider to use when sending the response. Applies only to SAML protocols
  - Name ID Format: This specifies the Name ID format that the service provider will request the identity provider to use when locating or creating a federation for the user. Applies only to SAML 2.0 protocol.
  - Requested Authentication Mechanism: This specifies the local authentication mechanism that the service provider will use. The service provider will map this local mechanism to a protocol-specific method, and specify this method in its authentication request to the identity provider. (See Section 5.14.1, "About Authentication Mechanisms".) Applies only to SAML 2.0/WS-Fed protocols.
  - Authentication Mechanism Comparison: If using SAML 2.0, specifies the comparator that the identity provider will use when determining the authentication mechanism to use. Options are:
    - `EXACT`: the identity provider must use the requested authentication mechanism
    - `MINIMUM`: the identity provider must use a mechanism that is at least as strong as the requested authentication mechanism
    - `BETTER`: the identity provider must use a mechanism that is stronger than the requested authentication mechanism
    - `MAXIMUM`: the identity provider must use a mechanism that is as strong as possible without exceeding the strength of the requested mechanism

3.2.7.3 Use the Test SP Engine with IdP-Initiated SSO

You can also use the test SP engine to test IdP-initiated Single Sign-On. In the service provider, simply enable the test SP engine and configure the default SP engine to be Test SP, and begin IdP-initiated SSO from the identity provider. The test SP engine will display the results of the Single Sign-On operation.

3.2.7.4 Test SP Engine Results

After Single Sign-On has been performed, the test SP engine displays the results of the operation, including:

- SSO Authentication Result: whether the operation was successful.
- User Identifier: the User ID of the user for which Single Sign-On was performed.
- Authentication Instant: the instant at which the identity provider authenticated the user.
- Session Expiration Instant: the instant at which the user session will become invalid.
- Authentication Mechanism: the local mechanism used to authenticate the user.
- SSO Primary Status Code: The primary status code in the assertion received from the identity provider.
- SSO Secondary Status Code: The secondary status code in the assertion received from the identity provider.
- SSO Status Message: The status message in the assertion received from the identity provider.
- IdP Provider ID: The Provider ID of the identity provider that authenticated the user.
- A list of the attributes from the assertion received from the identity provider, including:
  - `orafed-providerid`: The Provider ID of the identity provider that authenticated the user.
  - `orafed-nameid-format`: The format of the Name ID of the user federation.
  - `orafed-nameid-value`: The Name ID of the user federation.
  - `orafed-assertionid`: The ID of the assertion received from the identity provider.
  - Any user attributes included in the assertion received from the identity provider.

3.3 Post-Upgrade Administration

This section describes actions that the administrator must take following an upgrade of Oracle Identity Federation.

3.3.1 11g Server Signing Certificate

During an upgrade from release 10*g* to 11*g*, the upgrade assistant migrates the Oracle Identity Federation 10*g* Liberty/SAML2.0 signing key/certificate as the signing certificate of the Oracle Identity Federation 11*g* instance for use in SAML2.0/SAML1.x/WS-Fed operations.

If you used the Oracle Identity Federation 10*g* server to sign SAML 1.x/WS-Fed messages, you have two options:

- Keep the SAML1.x/WS-Fed key/certificate as the Oracle Identity Federation 11*g* signing key/certificate for all protocols (SAML2.0/SAML1.x/WS-Fed).

  To implement this, upload the 10*g* keystore as the 11*g* signing keystore. The 10*g* keystore can be found in the Oracle Identity Federation Administration Console by navigating to **SAML1x/WSFed**, then **Signer**; the 11*g* keystore is located in Fusion Middleware Control by navigating to the Oracle Identity Federation instance, in the **Security and Trust** section.

You must redistribute the SAML2.0 metadata to the remote partners after the change, so that the partners have the new signing certificate.

- Keep the SAML 2.0 signing key/certificate as the signing key/certificate of the Oracle Identity Federation 11*g* server.

  To implement this, you will need to provide the new signing certificate to all SAML 1.x/WS-Fed partners.

Part II

Administering Oracle Identity Federation

This part contains administration topics for Oracle Identity Federation.

Part II contains the following chapters:

4 Server Administration

This chapter describes tasks related to day-to-day administration of Oracle Identity Federation, and additional tasks that the administrator may need to perform on occasion. It contains these topics:

- Basic Administration
- Common Tasks
- Managing Identity Federations
- Configuring Identities
- Managing Credentials for Oracle Identity Federation

4.1 Basic Administration

This section describes basic administration of Oracle Identity Federation. It contains these topics:

- About the Oracle Identity Federation Server Administrator
- Administering Oracle Identity Federation
- Oracle Identity Federation Log Files
- Backups

4.1.1 About the Oracle Identity Federation Server Administrator

The Oracle Identity Federation administrator performs two major tasks, which can be characterized as:

- Basic runtime administration of the server, including starting, stopping, and monitoring the server
- Federated identity administration, which involves user administration (user creation, deletion, and federation), and maintaining information about trusted providers and the users affiliated with those providers

This section contains these topics:

- About Roles
- Deployment Planning
- Other Planning Tasks

4.1.1.1 About Roles

Oracle WebLogic Server defines certain global roles in the security realm that it installs, including:

- Admin (includes the Administrators group by default)
- Operator (includes the Operators group by default)
- Monitor (includes the Monitors group by default)

> **See Also:** Users, Groups, And Security Roles in *Oracle Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

The domain administrator can create the FederationAdmin role to grant Oracle Identity Federation administrator access to non-Admin/Operator/Monitor users.

> **Note:** The FederationAdmin role is recognized only by `WLST` commands and other JMX MBean clients, not by Fusion Middleware Control. This means that users with the FederationAdmin role will only be able to configure Oracle Identity Federation through `WLST` or other JMX clients, not through Fusion Middleware Control.

4.1.1.2 Deployment Planning

When deploying Oracle Identity Federation in a network of trusted sources and destinations, you will need to exchange information with other site administrators, and configure identity providers and service providers accordingly.

> **See Also:**
>
> - Chapter 5, "Configuring Oracle Identity Federation" for details about server properties
> - Section 4.3, "Managing Identity Federations" for details about trusted provider information

Exchange User Identities

In a federated environment, at the simplest level the service provider acts as a consumer of identity information, while the identity provider (where the user request originated) acts as the supplier of identity information. The identity provider may, in turn, adopt a consumer role as it communicates with an authentication and authorization mechanism (an AAA system) to obtain the necessary credentials. Service providers may also want to map users to identities at the destination, although this is not a requirement. Identity suppliers and consumers must be able to achieve a runtime exchange of data, which results in the source asserting some identity information about the principal which the destination can trust as a means of uniquely identifying the principal.

As an identity provider, you may wish to work with partner site administrators to provide the relevant lists of users from your domain. This is an optional information exchange.

Establish Cross-Domain Trust

Oracle Identity Federation can produce and consume provider metadata that conforms to the Liberty metadata specifications and to the SAML 2.0 metadata specifications.

Additionally, Oracle Identity Federation supports the ability to import provider metadata that uses the metadata extensions for SAML 2.0 query requesters.

You will need to establish cross-domain trust by setting up authentication and exchanging keys or certificates among the network of trusted sources and destinations.

For initial setup and testing, identity providers and service providers can both use default self-signed certificates. When going into production, however, consider the usage type when deciding whether self-signed certificates are sufficient: CA-issued certificates are most useful when there is no prior trust relation between entities, for example, when you use SSL to access a Web site over the Internet. But given that the trust relationship between federation peers requires the exchange of metadata or the equivalent, which can and usually does include the peer certificates, self-signed certificates should be sufficient for production deployment so long as you can trust how you obtained the peer certificates. Note that CA-issued certificates might be used in the metadata exchange, for example signed e-mail or a download from a web server over SSL.

The process of setting up cross-domain trust can be simplified by the use of metadata. Oracle Identity Federation enables you to store provider-specific metadata which overrides global IdP and SP settings with data specific to communication with each peer provider.

PKI and SSL Encryption

Oracle Identity Federation provides secure communication using X.509 certificate authentication.

Oracle Identity Federation provides encryption for data integrity using public key cryptography, a technique that uses a public and private key pair. Data is signed with a sending party's private key and the signature is verified by the recipient using the sender's public key.

Oracle Identity Federation uses documents known as **certificate**s to enable peer providers to establish trust. A **Certificate Authority (CA)** issues a certificate to vouch for a user's identity, including the party's public key in the certificate for use by the receiving party.

You configure key pairs and certificates using a local keystore. The identity provider configures a public and private key pair and a certificate - providing validation of the public key from a **Certificate Authority (CA)** - when using the POST profile. The presentation of the public key by the IdP, and certificate import by the SP, are critical aspects in managing the trust relationship between partners.

You can also implement SSL connections. For details on how to configure SSL connections and client certificates, see Section 8.1, "Configuring SSL for Oracle Identity Federation".

Note: SSL functionality is external to Oracle Identity Federation.

4.1.1.3 Other Planning Tasks

Besides exchanging identities and securing communications involving those identities, parties that plan to engage in a federated network must agree on a range of additional topics, such as:

- federation protocols
- services

- profiles

You will need to work with others in your network to ensure that the various IdPs and SPs understand their business partners' setups in order for federation to work properly.

4.1.2 Administering Oracle Identity Federation

You administer the Oracle Identity Federation server using the management tools in Oracle Fusion Middleware. See the following sections of the *Oracle Fusion Middleware Administrator's Guide* for details:

- Getting Started Using Oracle Enterprise Manager Fusion Middleware Control
- Getting Started Using Command-Line Tools
- Using the Fusion Middleware Control MBean Browsers

4.1.3 Oracle Identity Federation Log Files

Oracle Identity Federation log files are maintained in the `$DOMAIN_HOME/servers/servername/logs` directory and provide useful information for managing and monitoring server instances. The log files include:

Table 4–1 Oracle Identity Federation Log Files

Log File Name	Description
*servername*_`diagnostics`.log	Contains the runtime log records for the Oracle Identity Federation server.

Note: In prior releases, SAML messages exchanged between providers were maintained in `federation-msg.log`. This log file no longer exists; these messages are now audited and available in the Oracle Fusion Middleware Common Audit Framework.

4.1.4 Backups

You should back up your configurations/systems with the tools that you normally employ to back up your systems on a daily basis.

For more information about this topic, see Advanced Administration: Backup and Recovery in the *Oracle Fusion Middleware Administrator's Guide*.

Windows

Use this backup regimen:

- Use the backup/restore system tools on window platforms.
- Back up everything on all components in the Oracle Identity Federation configuration.

Linux/solaris:

Use this backup regimen:

- Shut down all Oracle WebLogic Server and Oracle Identity Federation components.

- Run the `tar` command on all components, including the Oracle Identity Federation folder under the managed server, and the RDBMS data files that Oracle Identity Federation is using.

  For example:

  ```
  tar cvzf oif11_backup oif_folder
  ```

4.2 Common Tasks

This section describes common services provided by Oracle Identity Federation for administrators and peer users. It explains these tasks:

- Obtain Server Metadata
- Obtain Server Certificates
- Perform SP-initiated Single Sign-On
- Perform IdP-initiated Single Sign-On
- Use the Relay State in IdP-initiated SSO
- Launch the Logout Process
- Set Signature Verification Certificate Property (SAML 1.x)
- Perform SP-initiated Single Sign-On (SAML 1.x)
- Send Attribute Requests and Queries (SAML 1.x)
- Send Authentication Queries (SAML 1.x)

4.2.1 Obtain Server Metadata

The Oracle Identity Federation metadata can either be retrieved from Oracle Enterprise Manager Fusion Middleware Control or by directly accessing a URL.

To retrieve the metadata from Fusion Middleware Control:

1. Navigate to **Oracle Identity Federation**, then **Administration**, then **Security and Trust**, then **Provider Metadata**.
2. Select the provider type and the version of the Oracle Identity Federation metadata to be created.
3. Click **Generate**.

To get the Oracle Identity Federation IdP metadata, go to a URL of the form:

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

To get the Oracle Identity Federation SP metadata, go to a URL of the form:

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

Sample IdP Metadata

The following is a sample of metadata for a server that has SSO Identity Provider, Attribute Authority, Authentication Query and Assertion ID Responder features enabled:

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
ID="id-PmHsOU3mD8zEyjDo0QbyelE5oxY-"
entityID="https://node.us.example.com:7002/fed/idp"
```

```
validUntil="2009-05-24T15:48:15Z">
   <md:IDPSSODescriptor WantAuthnRequestsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
      <md:KeyDescriptor use="signing">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICIzCCAYygAwIBAgIBJTANBgkqhkiG9w0BAQQFADA1MTMwMQYDVQQDEypz
dGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlmaWNhdGUwHhcNMDkwMTEzMjMwMTE2WhcNMT
AwMTEzMjMwMTE2WjA1MTMwMQYDVQQDEypzdGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlm
aWNhdGUwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAI7X7J6A057NEBgTnCYussaz6E3IY6JsgAYOiX
HfwunEv6zRZnpdVlZIRUyT+NNULSfk+PLbQU/NCg8yQdJeSNYkQ4BId+yyUDcYC447nhHa37uLKM7aWyAX
c6AeffC6CSEs0yZltgU2nIxJh9tLhPe5hzf0QjSImyXR/vjS/6nDAgMBAAGjQzBBMA8GA1UdEwEB/wQFMA
MBAf8wDwYDVR0PAQH/BAUDAwfwADAdBgNVHQ4EFgQUmZ8T7GkFv2VZB+FogX99DIvodTswDQYJKoZIhvcN
AQEEBQADgYEAbMGoZzjo9Bfaua3wiRh3LyMeahdoHv5S67JPAWNXrvQUxKjvYH0QR2oTnD+Rf3hIhi6Tjw
y4oP9YrcADiChp8tqckrBnR3L1aEErLXGau6r++a/PwslasuysNfbEoHrGJ1m+3K9DXGYYkGKdKgW9Dgg8
MObZshDxd7xUm557QO8=
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=node.us.example.com Signing
Certificate</dsig:X509IssuerName>
                  <dsig:X509SerialNumber>37</dsig:X509SerialNumber>
               </dsig:X509IssuerSerial>
               <dsig:X509SubjectName>CN=node.us.example.com Signing
Certificate</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
      </md:KeyDescriptor>
      <md:KeyDescriptor use="encryption">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICPDCCAeYCEC5V26OFPaoDxzAazNs8UBwwDQYJKoZIhvcNAQEEBQAweTEL
MAkGA1UEBhMCVVMxEDAOBgNVBAgTB015U3RhdGUxDzANBgNVBAcTBk15VG93bjEXMBUG
A1UEChMOTXlPcmdhbml6YXRpb24xGTAXBgNVBAsTEEZPUiBURVNUSU5HIE9OTFkxEzARBgNVBAMTCkNlcn
RHZW5DQUIwHhcNMDgxMTE4MjAwNzE4WhcNMjMxMTE5MjAwNzE4WjCBhTELMAkGA1UEBhMCVVMxEDAOBgNV
BAgWB015U3RhdGUxDzANBgNVBAcWBk15VG93bjEXMBUGA1UEChYOTXlPcmdhbml6YXRpb24xGTAXBgNVBA
sWEEZPUiBURVNUSU5HIE9OTFkxHzAdBgNVBAMWFnN0YTAwNTM0LnVzLm9yYWNsZS5jb20wgZ8wDQYJKoZI
hvcNAQEBBQADgY0AMIGJAoGBAMJCDgD00LDSUdWT0SznaU35ZkeQD2Ql6hvtoGcs8MfQpOM/yzM3C9GDlK
9+0JpN+7EFGQsCCezFVEX6lMzWdkvdGhbTUJ8/FI32QZ6FPkFItZrnfOS6eDpxcPsnv33rPVQ+ccRvj7BK
+sn24PEeV5rt3xF1cGuHGr57t/LtUa01AgMBAAEwDQYJKoZIhvcNAQEEBQADQQAChW8nbopN0FTyZRcVOT
ZKUlklHXf5X8Xi4gh2OIkkr7q9kjFlfI60SQZoD/nThn1sGZPPbtPGEwRevpqv7cI/
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=CertGenCAB, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509IssuerName>

<dsig:X509SerialNumber>61590287842211333696140797217026625564</dsig:X509SerialNumb
er>
               </dsig:X509IssuerSerial>
               <dsig:X509SubjectName>CN=node.us.example.com, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
         <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_
5"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>
```

```
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      </md:KeyDescriptor>
      <md:ArtifactResolutionService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://node.us.example.com:7002/fed/idp/soap" index="1"
isDefault="true"/>
      <md:SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://node.us.example.com:7002/fed/idp/samlv20"
ResponseLocation="https://node.us.example.com:7002/fed/idp/samlv20"/>
      <md:SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://node.us.example.com:7002/fed/idp/samlv20"
ResponseLocation="https://node.us.example.com:7002/fed/idp/samlv20"/>
      <md:SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://node.us.example.com:7002/fed/idp/samlv20ss"
ResponseLocation="https://node.us.example.com:7002/fed/idp/samlv20ss"/>
      <md:ManageNameIDService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://node.us.example.com:7002/fed/idp/samlv20"
ResponseLocation="https://node.us.example.com:7002/fed/idp/samlv20"/>
      <md:ManageNameIDService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://node.us.example.com:7002/fed/idp/samlv20"
ResponseLocation="https://node.us.example.com:7002/fed/idp/samlv20"/>
      <md:ManageNameIDService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://node.us.example.com:7002/fed/idp/samlv20ss"
ResponseLocation="https://node.us.example.com:7002/fed/idp/samlv20ss"/>
      <md:ManageNameIDService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://node.us.example.com:7002/fed/idp/soap"/>

<md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</md:NameIDFo
rmat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</md:NameIDFor
mat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName</md:Nam
eIDFormat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameID
Format>
      <md:SingleSignOnService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://node.us.example.com:7002/fed/idp/samlv20"/>
      <md:SingleSignOnService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://node.us.example.com:7002/fed/idp/samlv20"/>
      <md:SingleSignOnService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://node.us.example.com:7002/fed/idp/samlv20ss"/>
      <md:AssertionIDRequestService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://node.us.example.com:7002/fed/idp/soap"/>
      <md:AssertionIDRequestService
```

```
Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
Location="https://node.us.example.com:7002/fed/idp/assertionid"/>
   </md:IDPSSODescriptor>
   <md:AuthnAuthorityDescriptor
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
      <md:KeyDescriptor use="signing">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICIzCCAYygAwIBAgIBJTANBgkqhkiG9w0BAQQFADA1MTMwMQYDVQQDEypz
dGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlmaWNhdGUwHhcNMDkwMTEzMjMwMTE2WhcNMT
AwMTEzMjMwMTE2WjA1MTMwMQYDVQQDEypzdGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlm
aWNhdGUwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAI7X7J6A057NEBgTnCYussaz6E3IY6JsgAYOiX
HfwunEv6zRZnpdVlZIRUyT+NNULSfk+PLbQU/NCg8yQdJeSNYkQ4BId+yyUDcYC447nhHa37uLKM7aWyAX
c6AeffC6CSEs0yZltgU2nIxJh9tLhPe5hzf0QjSImyXR/vjS/6nDAgMBAAGjQzBBMA8GA1UdEwEB/wQFMA
MBAf8wDwYDVR0PAQH/BAUDAwfwADAdBgNVHQ4EFgQUmZ8T7GkFv2VZB+FogX99DIvodTswDQYJKoZIhvcN
AQEEBQADgYEAbMGoZzjo9Bfaua3wiRh3LyMeahdoHv5S67JPAWNXrvQUxKjvYH0QR2oTnD+Rf3hIhi6Tjw
y4oP9YrcADiChp8tqckrBnR3L1aEErLXGau6r++a/PwslasuysNfbEoHrGJ1m+3K9DXGYYkGKdKgW9Dgg8
MObZshDxd7xUm557QO8=
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=node.us.example.com Signing
Certificate</dsig:X509IssuerName>
                  <dsig:X509SerialNumber>37</dsig:X509SerialNumber>
               </dsig:X509IssuerSerial>
               <dsig:X509SubjectName>CN=node.us.example.com Signing
Certificate</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
      </md:KeyDescriptor>
      <md:KeyDescriptor use="encryption">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICPDCCAeYCEC5V26OFPaoDxzAazNs8UBwwDQYJKoZIhvcNAQEEBQAweTEL
MAkGA1UEBhMCVVMxEDAOBgNVBAgTB015U3RhdGUxDzANBgNVBAcTBk15VG93bjEXMBUG
A1UEChMOTXlPcmdhbml6YXRpb24xGTAXBgNVBAsTEEZPUiBURVNUSU5HIE9OTFkxEzARBgNVBAMTCkNlcn
RHZW5DQUIwHhcNMDgxMTE4MjAwNzE4WhcNMjMxMTE5MjAwNzE4WjCBhTELMAkGA1UEBhMCVVMxEDAOBgNV
BAgWB015U3RhdGUxDzANBgNVBAcWBk15VG93bjEXMBUGA1UEChYOTXlPcmdhbml6YXRpb24xGTAXBgNVBA
sWEEZPUiBURVNUSU5HIE9OTFkxHzAdBgNVBAMWFnN0YTAwNTM0LnVzLm9yYWNsZS5jb20wgZ8wDQYJKoZI
hvcNAQEBBQADgY0AMIGJAoGBAMJCDgD00LDSUdWT0SznaU35ZkeQD2Ql6hvtoGcs8MfQpOM/yzM3C9GDlK
9+0JpN+7EFGQsCCezFVEX6lMzWdkvdGhbTUJ8/FI32QZ6FPkFItZrnfOS6eDpxcPsnv33rPVQ+ccRvj7BK
+sn24PEeV5rt3xF1cGuHGr57t/LtUa01AgMBAAEwDQYJKoZIhvcNAQEEBQADQQAChW8nbopN0FTyZRcVOT
ZKUlklHXf5X8Xi4gh2OIkkr7q9kjFlfI60SQZoD/nThn1sGZPPbtPGEwRevpqv7cI/
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=CertGenCAB, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509IssuerName>

<dsig:X509SerialNumber>61590287842211333696140797217026625564</dsig:X509SerialNumb
er>
               </dsig:X509IssuerSerial>
               <dsig:X509SubjectName>CN=node.us.example.com, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
         <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_
5"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
```

```
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      </md:KeyDescriptor>
      <md:AuthnQueryService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://node.us.example.com:7002/fed/authnauth/soap"/>
      <md:AssertionIDRequestService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://node.us.example.com:7002/fed/authnauth/soap"/>
      <md:AssertionIDRequestService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
Location="https://node.us.example.com:7002/fed/authnauth/assertionid"/>

<md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</md:NameIDFo
rmat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName</md:Nam
eIDFormat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameID
Format>
   </md:AuthnAuthorityDescriptor>
   <md:AttributeAuthorityDescriptor
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
      <md:KeyDescriptor use="signing">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICIzCCAYygAwIBAgIBJTANBgkqhkiG9w0BAQQFADA1MTMwMQYDVQQDEypz
dGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlmaWNhdGUwHhcNMDkwMTEzMjMwMTE2WhcNMT
AwMTEzMjMwMTE2WjA1MTMwMQYDVQQDEypzdGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlm
aWNhdGUwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAI7X7J6A057NEBgTnCYussaz6E3IY6JsgAYOiX
HfwunEv6zRZnpdVlZIRUyT+NNULSfk+PLbQU/NCg8yQdJeSNYkQ4BId+yyUDcYC447nhHa37uLKM7aWyAX
c6AeffC6CSEs0yZltgU2nIxJh9tLhPe5hzf0QjSImyXR/vjS/6nDAgMBAAGjQzBBMA8GA1UdEwEB/wQFMA
MBAf8wDwYDVR0PAQH/BAUDAwfwADAdBgNVHQ4EFgQUmZ8T7GkFv2VZB+FogX99DIvodTswDQYJKoZIhvcN
AQEEBQADgYEAbMGoZzjo9Bfaua3wiRh3LyMeahdoHv5S67JPAWNXrvQUxKjvYH0QR2oTnD+Rf3hIhi6Tjw
y4oP9YrcADiChp8tqckrBnR3L1aEErLXGau6r++a/PwslasuysNfbEoHrGJ1m+3K9DXGYYkGKdKgW9Dgg8
MObZshDxd7xUm557QO8=
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=node.us.example.com Signing
Certificate</dsig:X509IssuerName>
                  <dsig:X509SerialNumber>37</dsig:X509SerialNumber>
               </dsig:X509IssuerSerial>
               <dsig:X509SubjectName>CN=node.us.example.com Signing
Certificate</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
      </md:KeyDescriptor>
      <md:KeyDescriptor use="encryption">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICPDCCAeYCEC5V26OFPaoDxzAazNs8UBwwDQYJKoZIhvcNAQEEBQAweTEL
MAkGA1UEBhMCVVMxEDAOBgNVBAgTB015U3RhdGUxDzANBgNVBAcTBk15VG93bjEXMBUG
A1UEChMOTXlPcmdhbml6YXRpb24xGTAXBgNVBAsTEEZPUiBURVNUSU5HIE9OTFkxEzARBgNVBAMTCkNlcn
RHZW5DQUIwHhcNMDgxMTE4MjAwNzE4WhcNMjMxMTE5MjAwNzE4WjCBhTELMAkGA1UEBhMCVVMxEDAOBgNV
```

```
BAgWB015U3RhdGUxDzANBgNVBAcWBk15VG93bjEXMBUGA1UEChYOTXlPcmdhbml6YXRpb24xGTAXBgNVBA
sWEEZPUiBURVNUSU5HIE9OTFkxHzAdBgNVBAMWFnN0YTAwNTM0LnVzLm9yYWNsZS5jb20wgZ8wDQYJKoZI
hvcNAQEBBQADgY0AMIGJAoGBAMJCDgD00LDSUdWT0SznaU35ZkeQD2Ql6hvtoGcs8MfQpOM/yzM3C9GDlK
9+0JpN+7EFGQsCCezFVEX6lMzWdkvdGhbTUJ8/FI32QZ6FPkFItZrnfOS6eDpxcPsnv33rPVQ+ccRvj7BK
+sn24PEeV5rt3xF1cGuHGr57t/LtUa01AgMBAAEwDQYJKoZIhvcNAQEEBQADQQAChW8nbopN0FTyZRcVOT
ZKUlklHXf5X8Xi4gh2OIkkr7q9kjFlfI60SQZoD/nThn1sGZPPbtPGEwRevpqv7cI/
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=CertGenCAB, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509IssuerName>

<dsig:X509SerialNumber>61590287842211333696140797217026625564</dsig:X509SerialNumb
er>
               </dsig:X509IssuerSerial>
               <dsig:X509SubjectName>CN=node.us.example.com, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
         <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_
5"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      </md:KeyDescriptor>
      <md:AttributeService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://node.us.example.com:7002/fed/aa/soap"/>
      <md:AssertionIDRequestService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://node.us.example.com:7002/fed/aa/soap"/>
      <md:AssertionIDRequestService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
Location="https://node.us.example.com:7002/fed/aa/assertionid"/>

<md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</md:NameIDFo
rmat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName</md:Nam
eIDFormat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameID
Format>

<md:AttributeProfile>urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic</md:Attr
ibuteProfile>
   </md:AttributeAuthorityDescriptor>
</md:EntityDescriptor>
```

Sample SP Metadata

The following is a sample of metadata for a server that has SSO Service Provider and Attribute Requestor features enabled:

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
ID="id-f4-F2z4ncIOsHw5w6CkMjneSE7I-"
entityID="http://node.us.example.com:7499/fed/sp"
```

```
validUntil="2009-05-24T15:39:48Z">
   <md:RoleDescriptor xmlns:query="urn:oasis:names:tc:SAML:metadata:ext:query"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" WantAssertionsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"
xsi:type="query:AttributeQueryDescriptorType">
      <md:KeyDescriptor use="signing">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICIzCCAYygAwIBAgIBJTANBgkqhkiG9w0BAQQFADA1MTMwMQYDVQQDEypz
dGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlmaWNhdGUwHhcNMDkwMTEzMjMwMTE2WhcNMT
AwMTEzMjMwMTE2WjA1MTMwMQYDVQQDEypzdGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlm
aWNhdGUwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAI7X7J6A057NEBgTnCYussaz6E3IY6JsgAYOiX
HfwunEv6zRZnpdVlZIRUyT+NNULSfk+PLbQU/NCg8yQdJeSNYkQ4BId+yyUDcYC447nhHa37uLKM7aWyAX
c6AeffC6CSEs0yZltgU2nIxJh9tLhPe5hzf0QjSImyXR/vjS/6nDAgMBAAGjQzBBMA8GA1UdEwEB/wQFMA
MBAf8wDwYDVR0PAQH/BAUDAwfwADAdBgNVHQ4EFgQUmZ8T7GkFv2VZB+FogX99DIvodTswDQYJKoZIhvcN
AQEEBQADgYEAbMGoZzjo9Bfaua3wiRh3LyMeahdoHv5S67JPAWNXrvQUxKjvYH0QR2oTnD+Rf3hIhi6Tjw
y4oP9YrcADiChp8tqckrBnR3L1aEErLXGau6r++a/PwslasuysNfbEoHrGJ1m+3K9DXGYYkGKdKgW9Dgg8
MObZshDxd7xUm557QO8=
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=node.us.example.com Signing
Certificate</dsig:X509IssuerName>
                  <dsig:X509SerialNumber>37</dsig:X509SerialNumber>
               </dsig:X509IssuerSerial>
               <dsig:X509SubjectName>CN=node.us.example.com Signing
Certificate</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
      </md:KeyDescriptor>
      <md:KeyDescriptor use="encryption">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICPDCCAeYCEC5V26OFPaoDxzAazNs8UBwwDQYJKoZIhvcNAQEEBQAweTEL
MAkGA1UEBhMCVVMxEDAOBgNVBAgTB015U3RhdGUxDzANBgNVBAcTBk15VG93bjEXMBUG
A1UEChMOTXlPcmdhbml6YXRpb24xGTAXBgNVBAsTEEZPUiBURVNUSU5HIE9OTFkxEzARBgNVBAMTCkNlcn
RHZW5DQUIwHhcNMDgxMTE4MjAwNzE4WhcNMjMxMTE5MjAwNzE4WjCBhTELMAkGA1UEBhMCVVMxEDAOBgNV
BAgWB015U3RhdGUxDzANBgNVBAcWBk15VG93bjEXMBUGA1UEChYOTXlPcmdhbml6YXRpb24xGTAXBgNVBA
sWEEZPUiBURVNUSU5HIE9OTFkxHzAdBgNVBAMWFnN0YTAwNTM0LnVzLm9yYWNsZS5jb20wgZ8wDQYJKoZI
hvcNAQEBBQADgY0AMIGJAoGBAMJCDgD00LDSUdWT0SznaU35ZkeQD2Ql6hvtoGcs8MfQpOM/yzM3C9GDlK
9+0JpN+7EFGQsCCezFVEX6lMzWdkvdGhbTUJ8/FI32QZ6FPkFItZrnfOS6eDpxcPsnv33rPVQ+ccRvj7BK
+sn24PEeV5rt3xF1cGuHGr57t/LtUa01AgMBAAEwDQYJKoZIhvcNAQEEBQADQQAChW8nbopN0FTyZRcVOT
ZKUlklHXf5X8Xi4gh2OIkkr7q9kjFlfI60SQZoD/nThn1sGZPPbtPGEwRevpqv7cI/
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=CertGenCAB, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509IssuerName>

<dsig:X509SerialNumber>61590287842211333696140797217026625564</dsig:X509SerialNumb
er>
               </dsig:X509IssuerSerial>
               <dsig:X509SubjectName>CN=node.us.example.com, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
         <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_
5"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
```

```
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      </md:KeyDescriptor>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName</md:Nam
eIDFormat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameID
Format>
      <md:NameIDFormat>lastname</md:NameIDFormat>
   </md:RoleDescriptor>
   <md:SPSSODescriptor AuthnRequestsSigned="true" WantAssertionsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
      <md:KeyDescriptor use="signing">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICIzCCAYygAwIBAgIBJTANBgkqhkiG9w0BAQQFADA1MTMwMQYDVQQDEypz
dGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlmaWNhdGUwHhcNMDkwMTEzMjMwMTE2WhcNMT
AwMTEzMjMwMTE2WjA1MTMwMQYDVQQDEypzdGEwMDUzNC51cy5vcmFjbGUuY29tIFNpZ25pbmcgQ2VydGlm
aWNhdGUwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAI7X7J6A057NEBgTnCYussaz6E3IY6JsgAYOiX
HfwunEv6zRZnpdVlZIRUyT+NNULSfk+PLbQU/NCg8yQdJeSNYkQ4BId+yyUDcYC447nhHa37uLKM7aWyAX
c6AeffC6CSEs0yZltgU2nIxJh9tLhPe5hzf0QjSImyXR/vjS/6nDAgMBAAGjQzBBMA8GA1UdEwEB/wQFMA
MBAf8wDwYDVR0PAQH/BAUDAwfwADAdBgNVHQ4EFgQUmZ8T7GkFv2VZB+FogX99DIvodTswDQYJKoZIhvcN
AQEEBQADgYEAbMGoZzjo9Bfaua3wiRh3LyMeahdoHv5S67JPAWNXrvQUxKjvYH0QR2oTnD+Rf3hIhi6Tjw
y4oP9YrcADiChp8tqckrBnR3L1aEErLXGau6r++a/PwslasuysNfbEoHrGJ1m+3K9DXGYYkGKdKgW9Dgg8
MObZshDxd7xUm557QO8=
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=node.us.example.com Signing
Certificate</dsig:X509IssuerName>
                  <dsig:X509SerialNumber>37</dsig:X509SerialNumber>
               </dsig:X509IssuerSerial>
               <dsig:X509SubjectName>CN=node.us.example.com Signing
Certificate</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
      </md:KeyDescriptor>
      <md:KeyDescriptor use="encryption">
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:X509Data>

<dsig:X509Certificate>MIICPDCCAeYCEC5V26OFPaoDxzAazNs8UBwwDQYJKoZIhvcNAQEEBQAweTEL
MAkGA1UEBhMCVVMxEDAOBgNVBAgTB015U3RhdGUxDzANBgNVBAcTBk15VG93bjEXMBUG
A1UEChMOTXlPcmdhbml6YXRpb24xGTAXBgNVBAsTEEZPUiBURVNUSU5HIE9OTFkxEzARBgNVBAMTCkNlcn
RHZW5DQUIwHhcNMDgxMTE4MjAwNzE4WhcNMjMxMTE5MjAwNzE4WjCBhTELMAkGA1UEBhMCVVMxEDAOBgNV
BAgWB015U3RhdGUxDzANBgNVBAcWBk15VG93bjEXMBUGA1UEChYOTXlPcmdhbml6YXRpb24xGTAXBgNVBA
sWEEZPUiBURVNUSU5HIE9OTFkxHzAdBgNVBAMWFnN0YTAwNTM0LnVzLm9yYWNsZS5jb20wgZ8wDQYJKoZI
hvcNAQEBBQADgY0AMIGJAoGBAMJCDgD00LDSUdWT0SznaU35ZkeQD2Ql6hvtoGcs8MfQpOM/yzM3C9GDlK
9+0JpN+7EFGQsCCezFVEX6lMzWdkvdGhbTUJ8/FI32QZ6FPkFItZrnfOS6eDpxcPsnv33rPVQ+ccRvj7BK
+sn24PEeV5rt3xF1cGuHGr57t/LtUa01AgMBAAEwDQYJKoZIhvcNAQEEBQADQQAChW8nbopN0FTyZRcVOT
ZKUlklHXf5X8Xi4gh2OIkkr7q9kjFlfI60SQZoD/nThn1sGZPPbtPGEwRevpqv7cI/
               </dsig:X509Certificate>
               <dsig:X509IssuerSerial>
                  <dsig:X509IssuerName>CN=CertGenCAB, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509IssuerName>
```

```
<dsig:X509SerialNumber>61590287842211333696140797217026625564</dsig:X509SerialNumb
er>
                </dsig:X509IssuerSerial>
                <dsig:X509SubjectName>CN=node.us.example.com, OU=FOR TESTING ONLY,
O=MyOrganization, L=MyTown, ST=MyState, C=US</dsig:X509SubjectName>
            </dsig:X509Data>
         </dsig:KeyInfo>
         <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_
5"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
         <md:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      </md:KeyDescriptor>
      <md:SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://node.us.example.com:7002/fed/sp/samlv20"
ResponseLocation="https://node.us.example.com:7002/fed/sp/samlv20"/>
      <md:SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://node.us.example.com:7002/fed/sp/samlv20"
ResponseLocation="https://node.us.example.com:7002/fed/sp/samlv20"/>
      <md:SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://node.us.example.com:7002/fed/sp/samlv20ss"
ResponseLocation="https://node.us.example.com:7002/fed/sp/samlv20ss"/>
      <md:ManageNameIDService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://node.us.example.com:7002/fed/sp/samlv20"
ResponseLocation="https://node.us.example.com:7002/fed/sp/samlv20"/>
      <md:ManageNameIDService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://node.us.example.com:7002/fed/sp/samlv20"
ResponseLocation="https://node.us.example.com:7002/fed/sp/samlv20"/>
      <md:ManageNameIDService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://node.us.example.com:7002/fed/sp/samlv20ss"
ResponseLocation="https://node.us.example.com:7002/fed/sp/samlv20ss"/>
      <md:ManageNameIDService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://node.us.example.com:7002/fed/sp/soap"/>
      <md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
Location="https://node.us.example.com:7002/fed/sp/art20" index="0"
isDefault="true"/>
      <md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://node.us.example.com:7002/fed/sp/authnResponse20" index="1"/>
      <md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://node.us.example.com:7002/fed/sp/authnResponse20ss" index="2"/>
      <md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS"
Location="https://node.us.example.com:7002/fed/sp/authnResponse20" index="4"/>
   </md:SPSSODescriptor>
</md:EntityDescriptor>
```

Sub-topics include:

- Versions
- Provider-specific Metadata

4.2.1.1 Versions

When issuing a metadata request for both the IdP and SP operations, you can specify the version of the metadata by specifying the version query parameter.

The query parameter must take one of these values:

- `saml10` - SAML 1.0 Metadata
- `saml11` - SAML 1.1 Metadata
- `saml20` - SAML 2.0 Metadata
- `lib11` - Liberty 1.1
- `lib12` - Liberty 1.2

Note: Liberty 1.x support is deprecated.

This example query requests IdP metadata for SAML 2.0 version:

```
http://host:port/fed/idp/metadata?version=saml20
```

Note: If the version parameter is missing, the version is assumed to be `saml20`.

4.2.1.2 Provider-specific Metadata

Oracle Identity Federation allows you to define global configuration parameters, and provides a way to override these global parameters when interacting with a specific remote provider. Some overridden parameters will result in a change of metadata.

For interoperability, Oracle Identity Federation supports metadata generation based on either the configuration of a specific provider, or by using the global settings.

If you choose to override the global configuration for a specific provider, you would typically generate the Oracle Identity Federation metadata based on the provider-specific configuration and provide it to the remote provider.

To generate the Oracle Identity Federation metadata for a specific remote provider, append the providerid query parameter to the URL, setting it to the ProviderID. For example:

```
http://host:port/fed/idp/metadata?providerid=idp.com
```

Note: If the `providerid` parameter is missing, Oracle Identity Federation generates its metadata based on the global configuration.

4.2.2 Obtain Server Certificates

To get Oracle Identity Federation's IdP certificate, go to a URL in the following format:

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

To get Oracle Identity Federation's SP certificate, go to a URL in the following format:

```
http://host:port/fed/sp/cert
```

Additional topics include:

- Specifying Certificate Usage
- Specifying Certificate Type

4.2.2.1 Specifying Certificate Usage

You can specify the use (signing or encryption) of the certificate to be returned by including the `use` parameter with one of these values:

- `enc` - encryption certificate
- `sign` - signing certificate

For example:

```
http://host:port/fed/idp/cert?use=enc
```

This request format returns the Oracle Identity Federation IdP's new encryption certificate.

Note: If you do not specify the `use` parameter, the signing certificate is returned by default.

4.2.2.2 Specifying Certificate Type

You can specify the type (new or old) of the certificate to be returned by including the type parameter with one of these values:

- `new` - new certificate
- `old` - old certificate

For example:

```
http://host:port/fed/idp/cert?type=old&use=enc
```

This request format returns the Oracle Identity Federation IdP's old encryption certificate.

Note: If you do not specify the type parameter, the new certificate is returned by default.

4.2.3 Perform SP-initiated Single Sign-On

A federation SSO operation can be initiated by directly requesting a service at the Oracle Identity Federation/SP instance

The URL to be requested on Oracle Identity Federation is of the form:

```
http://host:port/fed/sp/initiatesso
```

The following query parameters can be specified when requesting the URL:

- `providerid` - This is the identifier of the IdP to use to perform the SSO operation (optional). If missing, the default SSO provider is used.

- `federationid` - This is the identifier of the affiliation to use for the SSO (optional). See Section 6.2, "Working with Affiliations" for more information.
- `returnurl` - This is the URL to which the user is sent after a successful SSO operation. It is required if the Unsolicited Relay State property is empty.

An example of the URL format is:

```
http://host:port/fed/sp/initiatesso?providerid=http%3A%2F%2Fidp.com&returnurl=Prot
ectedAppURL
```

Note: Check that the query parameter values are correctly URL-encoded.

4.2.4 Perform IdP-initiated Single Sign-On

Oracle Identity Federation provides the ability to initiate an SSO operation by directly requesting a URL at the Oracle Identity Federation instance acting as an IdP; this is called an SSO IdP-initiated operation.

The URL to be requested on Oracle Identity Federation is of the form: `http://host:port/fed/idp/initiatesso`.

The following query parameters can be specified:

- `providerid` - This is the identifier of the SAML 2.0 SP to use to perform the SSO operation (optional).
- `federationid` - This is the identifier of the affiliation to use for the SSO (optional). See Section 6.2, "Working with Affiliations" for more information.
- `returnurl` - This is the URL to which the user is sent after a successful SSO operation.

An example of this type of URL is:

```
http://host:port/fed/idp/initiatesso?providerid=http%3A%2F%2Fsp.com&returnurl=Prot
ectedAppURL
```

Note: Check that the query parameter values are correctly URL-encoded.

4.2.5 Use the Relay State in IdP-initiated SSO

When an IdP performs unsolicited single sign-on operations, a relay state can be communicated to the service provider. The relay state can be used as a final target URL to which the user is sent after the Single Sign-On operation successfully ends.

The following scenarios are possible for IdP-initiated SSO:

Scenario 1: IdP-initiated SSO with Relay State

The IdP-initiated URL is of the form:

```
http://host:port/fed/idp/initiatesso?providerid=providerA&returnURL=http://somelin
k
```

In this case Oracle Identity Federation sees that there is a `returnURL` in the `initiatesso` directive and sets that to `relayState` to send to the SP.

Scenario 2: IdP-initiated SSO, SP Sets Relay State

The IdP-initiated URL is of the form:

```
http://host:port/fed/idp/initiatesso?providerid=providerA
```

In this case there is no `returnURL`, so the server looks at configuration for the SP (`providerA`). If there is a specific URL set there for unsolicited relay state, then it uses that URL to set as RelayState to send to the SP.

Scenario 3: IdP-initiated SSO, No Relay State

As in Scenario 2, the IdP-initiated URL is of the form:

```
http://host:port/fed/idp/initiatesso?providerid=providerA
```

In this case there is no returnURL, so the server looks at configuration for the providerA. If there is no URL set there for Unsolicited Relay State, then it does not send any RelayState to SP. In this scenario the server expects the SP to determine where to send the user after SSO is complete, based on their configuration/logic.

4.2.6 Launch the Logout Process

Launch the logout process by accessing a URL of the form:

```
http://hostname:port/fed/user/logout?returnurl=http%3A%2F%2Fanotherhostname%2Fpath
```

The logout service takes a `returnurl` parameter, which is necessary for correct operation; the user will be redirected to this URL after the logout process completes.

If no `returnurl` parameter is specified when invoking the Oracle Identity Federation logout URL, the sign-off operation is performed, and the server will display the built-in logout result page.

The logout is performed for all peer providers for the following protocols:

- SAML 2.0
- WS-Federation
- Liberty 1.1
- Liberty 1.2

For the SAML 1.x protocols, only the local Oracle Identity Federation session will be deleted. The SAML 1.x remote providers will not be notified of the user terminating the session.

See Also: Section 6.8.4, "Configuring On-Demand Global Logout" for details about configuring Oracle Identity Federation to enable the user to launch global logout.

4.2.7 Set Signature Verification Certificate Property (SAML 1.x)

If you did not upload XML metadata for a SAML 1.x provider, and that provider is configured to send signed messages, you need to do the following:

- If the provider is an IdP, you must set the `IdP Signature Verification Certificate` property.
- If the provider is an SP, you must set the `SP Signature Verification Certificate` property.

See Also: Section 4.2.2, "Obtain Server Certificates" for information on how to obtain these certificates.

4.2.8 Perform SP-initiated Single Sign-On (SAML 1.x)

Certain properties must be set before you can perform SP Initiated SSO.

After adding the SP metadata to the IdP, select it and click **Edit**. Set the following property in the **Trusted Provider Settings** section:

SP Assertion Consumer Service URL:

```
http://host:port/fed/sp/samlv11sso
```

After adding the IdP metadata to the SP, select it and click **Edit**. Set the following properties in the **Trusted Provider Settings** section:

- IdP Initiated SSO URL -- `http://host:port/fed/idp/samlv11sso` (for SAML 1.1) or `http://host:port/fed/idp/samlv10sso` (for SAML 1.0)
- IdP Initiated SSO Target Parameter -- any 'reasonable' string; used by the peer provider to identify the desired resource ("TARGET" in the case of Oracle Identity Federation).
- IdP Initiated SSO ProviderID Parameter - providerid
- IdP SOAP Artifact Resolution Endpoint -

  ```
  http://host:port/fed/idp/soapv11
  ```

4.2.9 Send Attribute Requests and Queries (SAML 1.x)

SAML 1.x defines a protocol for retrieving users' attributes. You can either send an attribute request to an Oracle Identity Federation instance acting as an SP, or send a SAML 1.x attribute query to an Oracle Identity Federation instance acting as an IdP.

Section 5.6, "Configuring Attribute Sharing with the Oracle Access Manager AuthZ Plug-in" shows how to configure an SP and IdP for attribute sharing for SAML.

To send an attribute request (in a SOAP envelope) to the SP, use the following URL:

```
http://sphost:port/fed/ar/soap
```

To send a SAML 1.x attribute query (in a SOAP envelope) to the IdP, use the following URL:

```
http://idphost:port/fed/aa/soapv11
```

For details, see:

- Section 5.6, "Configuring Attribute Sharing with the Oracle Access Manager AuthZ Plug-in"
- Section 4.3.5, "Set Up Single Sign-On for SAML 1.x and WS-Federation"

You can also configure attribute name and value mapping, and attribute filtering. See Section 5.9, "Configuring Attribute Mapping and Filtering" for details.

When Oracle Identity Federation, acting as an identity provider, receives a SAML 1.x attribute query, it will need to identify the requester. Oracle Identity Federation provides two ways in which a requester can be identified: by providing an SSL client certificate, or by authenticating with HTTP basic authentication.

If the requester authenticates by means of an SSL client certificate, the `cn` of the certificate subject must be the provider ID with which Oracle Identity Federation identifies the requester. If the requester authenticates through HTTP basic authentication, the username used by the requester must be the provider ID with which Oracle Identity Federation identifies the requester.

For details on how to set up SSL with client authentication or HTTP basic authentication, see Section 6.9, "Protecting the SOAP Endpoint".

4.2.9.1 NameID Format Strings when Using the Attribute Requester Service

The strings to be used for the `NameID` format when using the Attribute Requester service are:

Table 4–2 NameID Formats for Attribute Requester Service

Format	String
x509	oracle:security:nameid:format:x509
email	oracle:security:nameid:format:emailaddress
windows	oracle:security:nameid:format:windowsdomainqualifiedname
kerberos	oracle:security:nameid:format:kerberos
persistent	oracle:security:nameid:format:persistent
transient	oracle:security:nameid:format:transient
unspecified	oracle:security:nameid:format:unspecified
custom	oracle:security:nameid:format:custom
userid	oracle:security:nameid:format:userid

4.2.10 Send Authentication Queries (SAML 1.x)

To send SAML 1.x authentication queries to an Oracle Identity Federation instance acting as an IdP, use a URL of the form:

```
http://host:port/fed/authnauth/soapv11
```

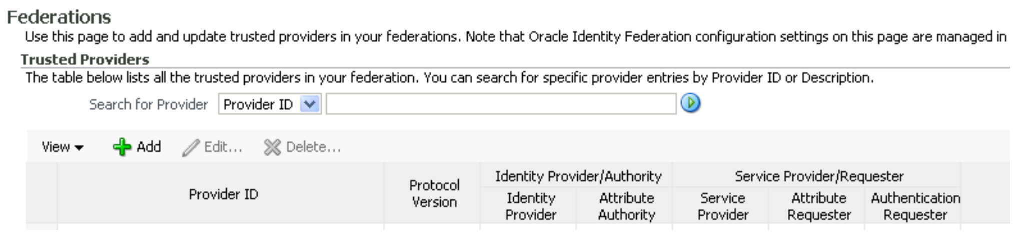
When Oracle Identity Federation, acting as an identity provider, receives a SAML 1.x authentication query, it must identify the requester. Oracle Identity Federation provides two ways in which a requester can be identified: by providing an SSL client certificate, or by authenticating through HTTP basic authentication.

If the requester authenticates using an SSL client certificate, the `cn` of the certificate subject must be the provider ID with which Oracle Identity Federation identifies the requester. If the requester authenticates through HTTP basic authentication, the username used by the requester must be the provider ID with which Oracle Identity Federation identifies the requester.

For details on how to set up SSL with client authentication or HTTP basic authentication, see Section 6.9, "Protecting the SOAP Endpoint".

4.3 Managing Identity Federations

You use Oracle Identity Federation server's Federations page in Fusion Middleware Control to view and manage the server's trusted providers.


Federations

Use this page to add and update trusted providers in your federations. Note that Oracle Identity Federation configuration settings on this page are managed in

Trusted Providers

The table below lists all the trusted providers in your federation. You can search for specific provider entries by Provider ID or Description.

Search for Provider: Provider ID

Provider ID	Protocol Version	Identity Provider/Authority		Service Provider/Requester		
		Identity Provider	Attribute Authority	Service Provider	Attribute Requester	Authentication Requester

To view the page in Fusion Middleware Control, select the instance of interest, and navigate to **Oracle Identity Federation**, then **Administration**, then **Federations**.

Topics in this section include:

- Search for a Provider
- Add Trusted Providers
- Update Trusted Providers
- Delete Trusted Providers
- Set Up Single Sign-On for SAML 1.x and WS-Federation

4.3.1 Search for a Provider

Use this feature to locate a provider from your trusted providers.

In the text box, enter the Provider ID, or the provider description. Click the search icon.

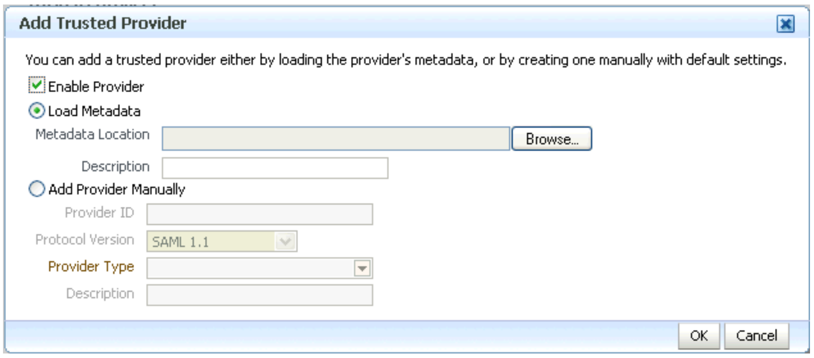
You can enter a partial Provider ID or description. For example, enter a port number to display only servers that listen on the specified port.

4.3.2 Add Trusted Providers

To add a trusted provider to your server's trusted providers:

1. In Fusion Middleware Control, select the server instance in the topology panel at left.
2. Navigate to Oracle Identity Federation, then **Administration**, then **Federations**.
3. On the **Federations** page, click **Add**.

   The Add Trusted Provider dialog appears. You can supply the provider details here or upload from the file system.


Add Trusted Provider

You can add a trusted provider either by loading the provider's metadata, or by creating one manually with default settings.

☑ Enable Provider

● Load Metadata

Metadata Location

Description

○ Add Provider Manually

Provider ID

Protocol Version: SAML 1.1

Provider Type

Description

Note the following points:

- When you upload metadata, there is no validation to check if the related SP or IdP site is using the same protocol. A federated single sign-on may be performed to validate the protocols.
- After using the Firefox browser to upload provider metadata from the Federations page in Fusion Middleware Control, you cannot modify the provider metadata file that you just uploaded until you restart Firefox.

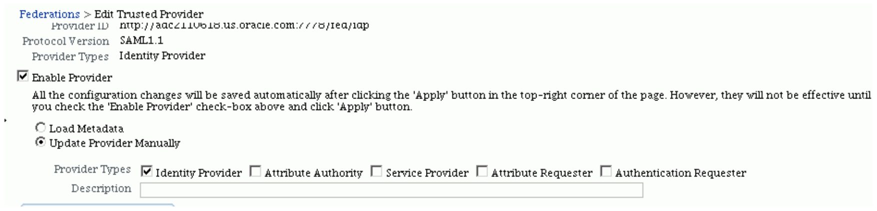
  The reason for this is the `Live HTTP Header` add-on in Firefox. Once you disable this add-on and restart Firefox, you can modify the provider metadata file after you have uploaded the metadata on the **Federations** page.

4.3.3 Update Trusted Providers

To update or delete a trusted provider:

1. In Fusion Middleware Control, select the server instance in the topology panel at left.
2. Navigate to Oracle Identity Federation, then **Administration**, then **Federations**.
3. Select the provider and click **Edit**.

The trusted provider control properties appear as follows:



Federations > Edit Trusted Provider

Provider ID	http://adc2110618.us.oracle.com:7778/fed/idp
Protocol Version	SAML1.1
Provider Types	Identity Provider

[✓] Enable Provider

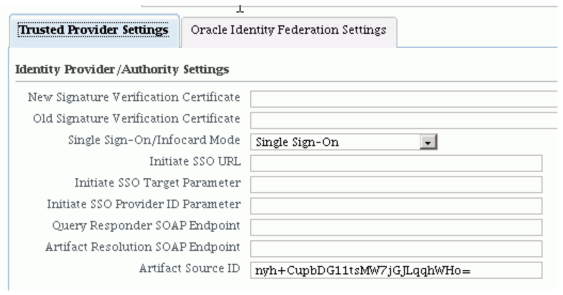
All the configuration changes will be saved automatically after clicking the 'Apply' button in the top-right corner of the page. However, they will not be effective until you check the 'Enable Provider' check-box above and click 'Apply' button.

○ Load Metadata
[● Update Provider Manually]

Provider Types: [✓] Identity Provider ☐ Attribute Authority ☐ Service Provider ☐ Attribute Requester ☐ Authentication Requester

Description

The Trusted Provider Settings tab appears as follows:

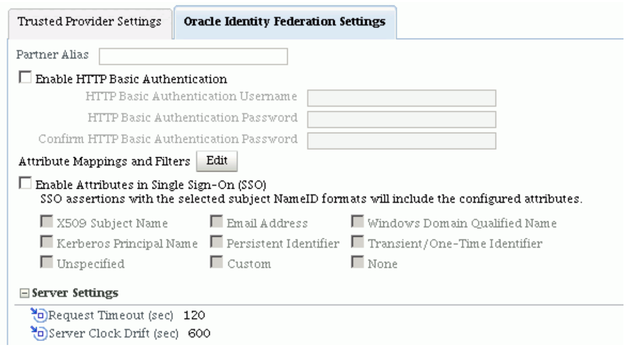


Trusted Provider Settings | Oracle Identity Federation Settings

Identity Provider/Authority Settings

New Signature Verification Certificate	
Old Signature Verification Certificate	
Single Sign-On/Infocard Mode	Single Sign-On
Initiate SSO URL	
Initiate SSO Target Parameter	
Initiate SSO Provider ID Parameter	
Query Responder SOAP Endpoint	
Artifact Resolution SOAP Endpoint	
Artifact Source ID	nyh+CupbDG11tsMW7jGJLqqhWHo=

The first set of fields on the Oracle Identity Federation Settings tab appears as follows:



Trusted Provider Settings | Oracle Identity Federation Settings

Partner Alias

☐ Enable HTTP Basic Authentication

HTTP Basic Authentication Username	
HTTP Basic Authentication Password	
Confirm HTTP Basic Authentication Password	

Attribute Mappings and Filters

☐ Enable Attributes in Single Sign-On (SSO)

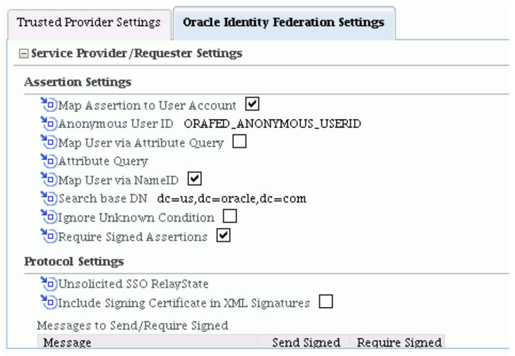
SSO assertions with the selected subject NameID formats will include the configured attributes.

☐ X509 Subject Name	☐ Email Address	☐ Windows Domain Qualified Name
☐ Kerberos Principal Name	☐ Persistent Identifier	☐ Transient/One-Time Identifier
☐ Unspecified	☐ Custom	☐ None

Server Settings

Request Timeout (sec)	120
Server Clock Drift (sec)	600

The second and final set of fields on the Oracle Identity Federation Settings tab appears as follows:



Trusted Provider Settings | **Oracle Identity Federation Settings**

Service Provider/Requester Settings

Assertion Settings

- Map Assertion to User Account ✓
- Anonymous User ID ORAFED_ANONYMOUS_USERID
- Map User via Attribute Query ☐
- Attribute Query
- Map User via NameID ✓
- Search base DN dc=us,dc=oracle,dc=com
- Ignore Unknown Condition ☐
- Require Signed Assertions ✓

Protocol Settings

- Unsolicited SSO RelayState
- Include Signing Certificate in XML Signatures ☐

Messages to Send/Require Signed

Message	Send Signed	Require Signed

See Also: Section 4.3.5, "Set Up Single Sign-On for SAML 1.x and WS-Federation".

4.3.4 Delete Trusted Providers

To update or delete a trusted provider from your server's trusted providers:

1. In Fusion Middleware Control, select the server instance in the topology panel at left.
2. Navigate to Oracle Identity Federation, then **Administration**, then **Federations**.
3. Select the provider and click **Delete**.

4.3.5 Set Up Single Sign-On for SAML 1.x and WS-Federation

To set up SSO for SAML 1.x and WS-Federation protocols, you first add the peer provider to the list of trusted providers, then fill in the required attributes for the protocol.

Add the Peer Provider to the Trusted Providers

Take these steps to achieve this task:

1. In Fusion Middleware Control, select the server instance in the topology panel at the left.
2. Navigate to **Oracle Identity Federation**, then **Administration**, then **Federations**.
3. On the Federations page, click **Add**.
4. Select the option to **Add Provider Manually**. Fill in the required attributes, and click **OK** to add the trusted provider.

Fill in the Required Attributes for SAML 1.x

You can choose to upload SAML1.x metadata, which can then be generated by Oracle Identity Federation. The peer provider's signing certificate and the artifact resolution SOAP endpoint parameters are automatically filled in.

Take these steps to achieve this task:

1. On the Federations page, select the SAML 1.X provider and click **Edit**.
2. Select the option to **Update Manually**.
3. Select the **Trusted Provider Settings** tab.

4. If the peer provider is an identity provider:

   a. In the Identity Provider / Authority Settings section enter the peer provider's signing certificate, in the **New Signature Verification Certificate** field.

   b. In the **Initiate SSO URL** field, enter the URL used by the peer provider to initiate the SSO flow.

   c. In the **Initiate SSO Target Parameter** field, enter the parameter used by the peer provider to identify the desired resource (`TARGET` in the case of Oracle Identity Federation).

   d. In the **Initiate SSO Provider ID Parameter** field, enter the name of the parameter that the peer provider uses to identify the provider ID of the service provider. For an Oracle Identity Federation peer identity provider, the parameter name must be "`providerid`".

   e. In the **Artifact Resolution SOAP Endpoint** field, enter the URL used by the peer provider to receive SOAP requests to resolve artifacts

   f. In the **Artifact Source ID** field, enter a source ID parameter. Oracle Identity Federation/SP uses this value to find the user's preferred identity provider from its list of known IdPs.

      If no source ID is specified, Oracle Identity Federation automatically generates it as the SHA-1 digest of the provider ID.

   For example, an Oracle Identity Federation peer identity provider might have the following Initiate SSO URL, using the configuration entered in steps b, c, and d above:

   ```
   http://saml.example.com/fed/idp/initiatesso?TARGET=<protected-resource-url>&pro
   viderid=<sp-provider-id-uri>
   ```

5. If the peer provider is a service provider:

   a. In the Service Provider / Requester Settings section enter the peer provider's signing certificate, in the **New Signature Verification Certificate** field.

   b. In the **Assertion Consumer Service URL** field, enter the URL used by the peer provider to consume the assertion.

Fill in the Required Attributes for WS-Federation SSO

Take these steps to achieve this task:

1. On the Federations page, select the WS-Fed 1.1 provider and click **Edit**.
2. Select the option to **Update Manually**.
3. Select the **Peer Provider Settings** tab.
4. If the peer provider is an identity provider:

   a. In the Identity Provider / Authority Settings section enter the peer provider's signing certificate, in **New Signature Verification Certificate** field.

   b. In the **Identity Realm Secure Token URL** field, add the URL of the peer provider to which the authentication request should be sent.

5. If the peer provider is a service provider:

   a. In the **Resource Realm Secure Token URL** field, enter the URL of the peer provider to which the assertion should be sent.

b. From the Assertion Type/Version drop-down list, select the version of the assertion that should be sent to this service provider.

Required Properties to Send to Peer Provider - WS-Federation

As IdP needs to send to peer provider (SP)

- ProviderID `http(s)://server_name:server_port/fed/idp`
- Signing Certificate
- Identity Realm Secure Token URL (The URL to which authentication requests should be sent `http(s)://`*`server_name:server_port`*`/fed/idp/wsfed11`)

As SP, needs to send to peer provider (IdP)

- ProviderID `http(s)://`*`server_name:server_port`*`/fed/sp`
- Resource Realm Secure Token URL (The URL to which assertions should be sent `http(s)://`*`server_name:server_port`*`/fed/sp/wsfed11)`
- Assertion Type/Version (The version of the assertion to be sent; optional)

Required Properties to Send to Peer Provider - SAML 1.x

The IdP needs to send to peer provider (SP)

- ProviderID `http(s)://`*`server_name:server_port`*`/fed/idp`
- Signing Certificate
- IdP initiate SSO URL (the URL to which authentication requests should be sent)

  For SAML 1.1:

  `http(s)://`*`server_name:server_port`*`/fed/idp/samlv11sso`

  For SAML 1.0:

  `http(s)://`*`server_name:server_port`*`/fed/idp/samlv10sso`
- IdP Artifact resolution URL (URL for sending artifact query) `http(s)://`*`server_name:server_port`*`/fed/idp/soapv11`
- Target URL query parameter (for Oracle Identity Federation it is TARGET). The parameter contains the URL on which the user lands after a successful sign-on.
- ProviderID URL query parameter (for Oracle Identity Federation it is providerid). The parameter contains the SP providerID when SP initiates a SAML1.x SSO
- SourceID required for the artifact profile. The value is obtained from the IdP metadata, from tag `<saml1md:SourceID>`

As SP, needs to send to peer provider (IdP):

- ProviderID `http(s)://`*`server_name:server_port`*`/fed/sp`
- Signing Certificate (if signing SOAP requests)
- Assertion Consumer URL (The URL to which assertions should be sent `http(s)://`*`server_name:server_port`*`/fed/sp/samlv11sso`)

4.4 Configuring Identities

Use this page to locate and manage user and federated identities, and to maintain search options.

- About Federated Identities
- Identities - Federations
- Identities - Users
- Identities - Search Options

Note: Liberty 1.x support is deprecated.

4.4.1 About Federated Identities

When a user performs SSO for the first time, and Oracle Identity Federation is configured with a federation data store, it creates a federation record for the user containing information about the federated identity. This includes:

- the Username – the User ID with which Oracle Identity Federation identifies the user.
- the User description – the user description
- the IdP Provided Name ID – the Name ID provided by the identity provider when SSO was performed
- the IdP Provided Name ID Format – the format of the Name ID provided by the identity provider. This field will be empty if the format is "persisent".
- the IdP Provided Name ID Qualifier - the Name ID qualifier, optionally provided by the identity provider.
- the Protocol Version – the protocol version used when performing SSO.
- the SP Provided Name ID – the Name ID provided by the service provider. This will be blank when the federation is created.
- the SP Provided Name ID Format – the format of the Name ID provided by the service provider. This field will be empty if the format is "persisent".
- the SP Provided Name ID Qualifier – the Name ID qualifier, optionally provided by the service provider.
- the SP Provided Name ID Version - the version of the SP Name ID.
- the Provider ID – the provider ID of the peer provider with which SSO was performed.
- the Federation Type – the type of federation that was created. It can have one of these values:
  - 1 - federation between this server as an IdP and an SP
  - 2 - federation between this server as an IdP and an Affiliation
  - 3 - federation between this server as an SP and an IdP
  - 4 - federation between Oracle Internet Directory server as an Affiliation and an IdP.

When the identity provider updates a federation by performing a Manage Name ID (MNI) operation, the value of the following fields is updated:

- the IdP Provided Name ID
- the IdP Provided Name ID Format
- the IdP Provided Name ID Qualifier

- the Protocol Version

When the service provider updates a federation by performing a Manage Name ID (MNI) operation, the value of the following fields is updated.

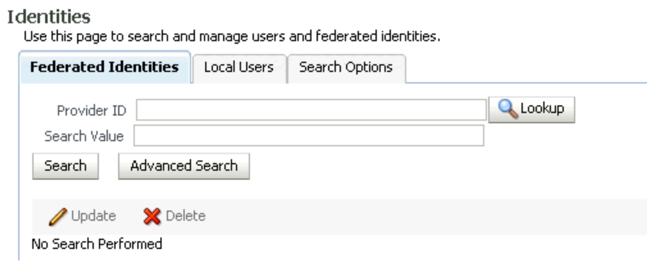
- the SP Provided Name ID
- the SP Provided Name ID Format
- the SP Provided Name ID Qualifier
- the SP Provided Name ID Version

The following fields will be empty when the federation record is created:

- the SP Provided Name ID Format.
- the SP Provided Name ID Qualifier.
- the SP Provided Name ID Version.

4.4.2 Identities - Federations

Use this page to locate and maintain federated identity records.



Simple Search

The following search fields are provided:

- Provider ID: Enter a Provider ID and click **Lookup** to choose the correct ID from a list of trusted providers. Entering a Provider ID in this field will limit the search to only those federated identities created with the provider specified. If no Provider ID is specified, the search will be performed over the federated identities created with all trusted providers.
- Search Value: Enter the value you wish to search for. If no value is specified, all federated identity records will be returned.

The search returns a table of federation records. The table columns appearing in the table depend on the default display attributes configured in the search options configuration.

See Also: Section 4.4.4, "Identities - Search Options"

Advanced Search

Take these steps to perform an advanced search:

- Enter a **Provider ID**, or part of one and click **Lookup** to choose the correct ID from a list of trusted providers. Entering a provider ID in this field will limit the search to only those federated identities created with the provider specified. If no Provider ID is specified, the search will be performed over the federated identities created with all trusted providers.

- Check **Include New and Old Name IDs in Search** to include the new and old NameID values in the search for federation records.

> **Note:** The new and old NameID fields are populated only if an update operation was previously performed on a federation record, and if the protocol is enabled.
>
> If NameID registration is disabled or if no update operation was ever performed, there is no need to include those fields during a search operation.

- Use the Operator radio buttons to specify whether the returned records must satisfy all conditions (`And`) or records satisfying any conditions (`Or`).
- Add attribute search conditions by following these steps:
  - Click **Add Attribute**.
  - A pop-up box appears. Use the drop-down list to select a federated identity attribute, and click **OK**.
  - The attribute appears as a search option. Select the comparator to use and the value to search for.
- Click **Search**.

To Manage Records

To manage a displayed record, select the corresponding row. Buttons on the page provide these actions:

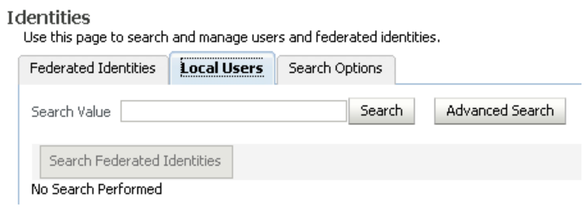
- **Update** - Updates the Name ID of the federated identity by performing a Manage Name ID (MNI) operation. See Section 4.4.1, "About Federated Identities" for more details.
- **Delete** - Terminates the federated identity by performing a Manage Name ID (MNI) operation with the "Terminate" flag set to `true`, and deletes the record.

The functions are available for the SAML 2.0 and Liberty 1.x protocols.

> **Note:** Liberty 1.x support is deprecated.

4.4.3 Identities - Users

Use this page to locate and maintain user records.



Simple Search

Enter the search value for which you wish to search in the local user entries, and click **Search**. If no value is specified, all users will be returned.

The search returns a table of user records. The table columns appearing in the table depend on the default display attributes configured in the search option configuration.

See Also: Section 4.4.4, "Identities - Search Options"

Advanced Search

Take these steps to perform an advanced search:

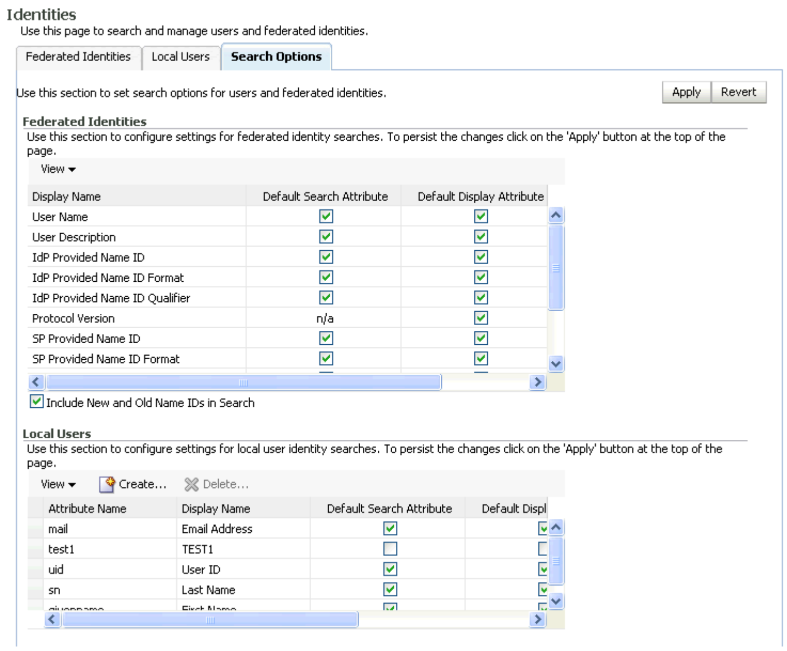
- Use the Operator radio buttons to specify whether the returned records must satisfy all conditions (`And`) or records satisfying any conditions (`Or`).
- Add attribute search conditions by following these steps:
  - Click **Add Attributes**.
  - A pop-up box appears. Select a user attribute from the list, and click **OK**.

    Note: The attributes that appear in this list are those configured in the search options configuration Section 4.4.4, "Identities - Search Options".

  - The attribute appears as a search condition. Select the comparator to use and the value to search for.
  - Click **Search**.

4.4.4 Identities - Search Options

Use this page to configure the attributes used to search for users and federation records.



Use this page to search and manage users and federated identities.

Use this section to set search options for users and federated identities.

Federated Identities

Use this section to configure settings for federated identity searches. To persist the changes click on the 'Apply' button at the top of the page.

Display Name	Default Search Attribute	Default Display Attribute
User Name	✓	✓
User Description	✓	✓
IdP Provided Name ID	✓	✓
IdP Provided Name ID Format	✓	✓
IdP Provided Name ID Qualifier	✓	✓
Protocol Version	n/a	✓
SP Provided Name ID	✓	✓
SP Provided Name ID Format	✓	✓

[✓] Include New and Old Name IDs in Search

Local Users

Use this section to configure settings for local user identity searches. To persist the changes click on the 'Apply' button at the top of the page.

Attribute Name	Display Name	Default Search Attribute	Default Displ
mail	Email Address	✓	✓
test1	TEST1	☐	☐
uid	User ID	✓	✓
sn	Last Name	✓	✓

The page displays two tables:

- The Federations table shows the attributes available on the "Identities - Federation Records" tab of the Identities page. It shows:

  - the attribute's display name
  - whether this attribute is a default search attribute in simple searches; in other words, whether a simple search will be performed over this attribute.
  - whether this attribute is displayed by default in the results from simple and advanced searches.

  Use the check boxes to specify which attributes should be available by default for search and display, respectively. Click **Apply** to save your changes.

- The Local Users table shows the attributes available on the **Identities - Users** tab of the Identities page. It shows:
  - the attribute's name in the user data store
  - the attribute's display name
  - whether this attribute is a default search attribute in simple searches; in other words, whether a simple search will be performed over this attribute
  - whether this attribute is displayed by default in the results from simple and advanced searches.

  Use the checkbox to specify which attributes should be available by default for search and display, respectively. Click **Apply** to save your changes.

  Click **Create** to add another attribute to the list. Enter the following values:
  - Attribute Name - The attribute name in the user data store
  - Display Name – The name that will be displayed when referencing this attribute
  - Default Search Attribute – Check if this attribute should be searched over in simple searches.
  - Default Display Attribute – Check if this attribute should be displayed in the search results.
  - Sort On – Check this if search results should be sorted based on the value of this attribute.

  Select an attribute and click **Delete** to remove it from the list of attributes available for display and searches. Once you delete an attribute, it is no longer available, but you can add it back to the list using **Create**.

Attributes for Federated Identities

The available attributes are:

- User Name
- User Description
- IdP ID
- IdP Format
- IdP Qualifier
- Protocol Version
- SP Provider ID
- SP Provider ID Format
- SP Provider ID Version

- Provider ID
- Federation Type

  The federation type can have these values:

  – 1 - federation between this server as an IdP and an SP
  – 2 - federation between this server as an IdP and an Affiliation
  – 3 - federation between this server as an SP and an IdP
  – 4 - federation between Oracle Internet Directory server as an Affiliation and an IdP

Check **Include New and Old Name IDs in Search** to include the new and old NameID values in the search operation for federation records.

The new and old NameID fields are populated only if an update operation was performed on a federation record, and if the NameID registration protocol is enabled.

If the protocol is disabled or if no update operation was ever performed, there is no need to include those fields during a search operation.

Attributes for Users

Attributes for users need to be added in the Local Users table in the Search Options tab before searches for local users can be performed. However, if the User data store is of type LDAP, the following attributes have already been added:

- Email Address
- User ID
- Last Name
- First Name

4.5 Managing Credentials for Oracle Identity Federation

Several properties used by Oracle Identity Federation are stored in the credential store. Table 4–3 provides the list of properties:

Note: Quoted values are the literal values stored.

Table 4–3 Oracle Identity Federation Properties Stored in Credential Store

Key	Description	Username	Password	Alias
jcepwdsign	PKCS#12/JCE signing password	"UniqueUserNameCredential"	*password*	*ApplicationName*
oldjcepwdsign	Old PKCS#12/JCE signing password	"UniqueUserNameCredential"	*password*	*ApplicationName*
jcepwdenc	PKCS#12/JCE encryption password	"UniqueUserNameCredential"		*ApplicationName*
oldjcepwdenc	Old PKCS#12/JCE encryption password	"UniqueUserNameCredential"	*password*	*ApplicationName*
userldappassword	LDAP credentials for user data store	User Store Bind DN (userldapusername)	*password*	*ApplicationName*
fedldappassword	LDAP credentials for fed data store	Fed Store Bind DN (fedldapusername)	*password*	*ApplicationName*

Table 4–3 (Cont.) Oracle Identity Federation Properties Stored in Credential Store

Key	Description	Username	Password	Alias
ldappassword	LDAP credentials for LDAP Authn Engine	Auth Engine Bind DN (ldapbinddn)	*password*	*ApplicationName*
ossopartnerkey	Oracle SSO partner key	"UniqueUserNameCredential"	*password*	*ApplicationName*
ossooldpartnerkey	Old Oracle SSO partner key	"UniqueUserNameCredential"	*password*	*ApplicationName*
proxypassword	Proxy password in Server Properties page of Fusion Middleware Control	Proxy User Name (proxyuser)	*password*	*ApplicationName*
PROVIDER_ID	Password for HTTP Basic Authn in the Peer Provider specific page	HTTP Basic Authn Username (authnbasicusername)	*password*	*ApplicationName*

Note that:

- the password is entered by the user during configuration
- *ApplicationName* is the name with which Oracle Identity Federation is deployed; by default this is *OIF*.
- `Authn Engine Bind DN` is the Bind DN specified by user for the LDAP authentication Engine
- `Fed Store Bind DN` is Bind DN specified by user for the LDAP federation store
- `User Store Bind DN` is the Bind DN specified by user for the user identity store.

Managing Credentials Manually

Note: When executing these `WLST` commands, connect to the administration server instead of the managed server where Oracle Identity Federation is running.

Credentials are automatically stored in the credential store when you configure Oracle Identity Federation through Fusion Middleware Control.

If needed, you can update these credentials manually using `WLST` commands or using Fusion Middleware Control.

See Also: Configuring the Credential Store in the *Oracle Fusion Middleware Application Security Guide* for configuration details using both these tools.

As an example of using the interactive WLST mode, you can issue the following commands to store `userldappassword` in the credential store so that it is accessible to Oracle Identity Federation.

This example assumes that the LDAP bind `dn` is `cn=orcladmin`, the correct password is substituted for *password*, and Oracle Identity Federation is deployed with application name `OIF`:

Create the `userldappassword` credential:

```
createCred(map="OIF", key="userldappassword",
user="cn=orcladmin", password="password", desc="user ldap password")
```

Update the `userldappassword` credential:

```
updateCred(map="OIF", key="userldappassword",
user="cn=orcladmin", password="password", desc="User ldap password")
```

Delete the `userldappassword` credential:

```
deleteCred(map="OIF", key="userldappassword")
```

5

Configuring Oracle Identity Federation

This chapter describes configuration tasks for Oracle Identity Federation. It contains these topics:

5.1 Data Maintained by Oracle Identity Federation

The Oracle Identity Federation administrator acquires the data needed to manage and operate the server from a variety of sources, including third parties (other providers' administrators), agreements with the third parties, and from local configuration decisions. The administrator is responsible for loading and maintaining this information in the federation server.

Broadly speaking, the federation server maintains two categories of configuration details:

- Server Configuration Data, which includes properties that determine the runtime behavior of a federation server instance
- User Federation Data, including details about individual users' federated identities and usage information

Note: Liberty 1.x support is deprecated.

5.1.1 Server Configuration Data

Each Oracle Identity Federation instance maintains two types of configuration data:

- Protocol data, including:
  - properties of the server instance as a whole, including the hostname and port, whether SSL is enabled, signing and encryption PKCS#12/JKS keystores, and so on
  - how the server instance supports its enabled federation protocols when acting as an identity provider, including session time-outs, re-authentication time-outs, the default provider ID, and so on
  - how the server instance supports its enabled federation protocols when acting as a service provider. The data maintained in this case is very similar to the data stored when the server acts as an identity provider
- Information about peer providers that are trusted providers of this server. Trusted provider configuration data includes:
  - name ID formats to use for assertions
  - attributes to send along with an authentication response
  - signing requirements for assertions and authentication requests
  - preferred bindings
  - validity periods of assertions and artifacts
  - other time-related parameters such as the allowable time difference between servers that are not synchronized.
  - account linking parameters

Configuration Settings and Provider Metadata

Note that relationships may exist between configuration settings and the provider metadata that the server generates. Some settings do not affect the metadata while others do. For example, changing the Session Timeout value does not affect the metadata, but changing the SOAP port will require the administrator to re-publish his metadata to the other trusted providers. Likewise, the administrator must be aware of changes to peer providers' metadata.

Here is a list of properties that affect metadata:

- Metadata Properties
  - Signing Metadata
  - Validity Period
- Server Properties
  - Server Hostname
  - Server Port
  - SOAP Port
  - IdP Enabled
  - SP Enabled

  - SSL Enabled
  - Signing PKCS #12/JKS Keystore
  - Encryption PKCS #12/JKS Keystore
- Common IdP Properties
  - ProviderID
  - SAML 2.0 Enabled
- Common SP Properties
  - ProviderID
  - SAML 2.0 Enabled
  - Enable Attribute Requester Service
- SAML 2.0 IdP Properties
  - Enable Protocol Profiles
  - Federation Termination Enabled
  - Register NameID Enabled
  - Attribute Responder Enabled
- SAML 2.0 SP Properties
  - Enable Protocol Profiles
  - Federation Termination Enabled
  - Register NameID Enabled

The metadata URLs for the various protocols are in this format:

> **Note:** You can retrieve the metadata from Fusion Middleware Control, by navigating to **Security and Trust**, then **Provider Metadata**.

- IdP metadata URL - `http(s)://hostname:port/fed/idp/metadata?version=version`
- SP metadata URL - `http(s)://hostname:port/fed/sp/metadata?version=version`

where version can be saml20, saml11, saml10, lib11, or lib12.

5.1.2 User Federation Data

A data store contains each user's identity federation data; the data store can be an LDAP directory or an RDBMS. In addition to the user's basic reference information, there are records for each unique identity federation associated with the user. A federation record is defined by:

- the remote provider
- the name identifier type (for example, an e-mail address or a DN)
- the protocol (for example, SAML 2.0)

This means that a user can have multiple identity federation records for the same remote provider, so long as the combination of these three attributes provides uniqueness. For example, the user's first record could be identified by a combination of ProviderX/myemail1/SAML 2.0, and the second record by ProviderX/myemail2/SAML 2.0.

Synchronization

As mentioned earlier, the federation records for a user are stored independently, and rely on a unique user attribute (such as a DN or a username) to link to the user record in the user data store.

An event that changes a user's unique attribute value - for example, if an employee moves to a new office location and her DN is updated - requires that the user's federations be dropped and re-established.

If a user's attribute value in the user store has changed, the user's federation record can be updated, for example in Fusion Middleware Control, from the **Identities** page.

Deprovisioning

Likewise, if a user record is deleted, the federation data remains. This means that the administrator must be sure to delete the user's federation data when the user is de provisioned.

Caution: Failure to delete the federation data in this situation can introduce a potential security problem. For example, consider a scenario where a new user is subsequently provisioned with the same unique attribute value - for example, the same DN or username; that user would inherit the previous user's account linkages if they had been left around.

The federation data can be deleted:

- using the LDAP server's or database's administration tools. For data stored in Oracle Internet Directory, see *Oracle Identity Management User Reference* to obtain more information.
- using a command-line utility provided with Oracle Identity Federation. For details, see Chapter 9, "Oracle Identity Federation Command-Line Tools".

5.2 Configuring Server Properties

Server properties include:

- Host Connection Properties
- Outbound Connection Properties

5.2.1 Host Connection Properties

These types of properties are configured for the server:

- basic connection parameters
- encryption settings
- logout options

Connection Parameters

You can configure the following parameters:

- Host

  This is the host name of the Oracle Identity Federation instance.

  > **Note:** This property affects server metadata. When updating this property, distribute the updated metadata to all your trusted providers.

  If there is a change to the host or port of the server, you can either define a virtual hostname or proxy server hostname, or else change the server host property.

- Port

  This is the port where Oracle Identity Federation listens.

  > **Note:**
  >
  > - This setting only dictates what server port will be specified in the IdP and SP metadata when the metadata is generated. If there are several HTTP or HTTPS ports enabled for the container instance in which Oracle Identity Federation is running, a user or peer provider can access Oracle Identity Federation through any of those ports, not just the port you specify here.
  > - This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.

  Checking the `SSL Enabled` box enables Secure Sockets Layer (SSL) encryption, allowing the server to listen in HTTPS mode.

  > **Note:**
  >
  > - This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.
  > - This setting only dictates what protocol (`http` or `https`) will be specified in the IdP and SP metadata when the metadata is generated. Setting this property does not configure SSL. For details of how to enable SSL, see:
  >   - Section 8.1, "Configuring SSL for Oracle Identity Federation"
  >   - *Oracle Fusion Middleware Administrator's Guide*

  Checking the **Force SSL** box forces communications with the server to be conducted in HTTPS mode. If true, Oracle Identity Federation checks an incoming connection to ensure that it is done over SSL. If it is not, the server redirects the user to a URL supporting SSL; the URL is built with the host name and port properties and the requested URL.

- SOAP Port

This is the port where Oracle Identity Federation listens for SOAP messages.

Note:

- This setting only dictates what SOAP port will be specified in the IdP and SP metadata when the metadata is generated. If there are several HTTP or HTTPS ports enabled for the container instance in which Oracle Identity Federation is running, a user or peer provider can access Oracle Identity Federation through any of those ports, not just the port you specify here.
- This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.

Checking the `SSL Enabled` box enables Secure Sockets Layer (SSL) encryption, allowing the server to listen in HTTPS mode.

Note:

- This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.
- Setting this property does not configure SSL. For details of how to enable SSL, see:
  - Section 8.1, "Configuring SSL for Oracle Identity Federation"
  - *Oracle Fusion Middleware Administrator's Guide*

Checking the **Force SSL** box forces communications with the server to be conducted in HTTPS mode. If true, Oracle Identity Federation checks an incoming connection to ensure that it is done over SSL. If it is not, the server redirects the user to a URL supporting SSL; the URL is built with the host name and port properties and the requested URL.

Checking `Require Client Certificate` forces SSL client authentication in all incoming SOAP connections.

- Server Clock Drift

  This is the allowable time difference, in seconds, between Oracle Identity Federation and its peer servers. The default is 600 seconds.

- Session Timeout

  This parameter is used to determine the period, in seconds, for which an authenticated session is active. If the session remains inactive beyond the active period, the user must re-authenticate. The default value is 7200 seconds.

  How this parameter is used depends on the server's role and the nature of the session in question.

  Scenario 1: User Authenticated Locally

  The user can be authenticated locally when:

  - Oracle Identity Federation acts as an IdP

- Oracle Identity Federation is an SP, and the user needs to be prompted for its credentials because a new federation is being created

In this case, the expiration time of the authenticated session is set to the value of the Session Timeout parameter.

Scenario 2: Existing Federation

When Oracle Identity Federation is acting as an SP with an existing federation, the server receives a SAML assertion from the IdP containing user and authentication information. The assertion may include a `ReauthenticateOnOrAfter` attribute, indicating to Oracle Identity Federation that the user should be re-authenticated after the period specified by the attribute.

In this case, the Oracle Identity Federation server acting as SP sets the expiration time of the authenticated session to: the `Session Timeout` parameter or the `ReauthenticateOnOrAfter` assertion attribute, whichever is less.

Note: When Oracle Identity Federation uses Oracle Access Manager or Oracle Single Sign-On as its user data store, the Session Timeout has no effect on the user session. With Oracle Access Manager, the session timeout is determined by the configuration of the AccessGates protecting accessed resources.

- Request Timeout

  This is the validity time, in seconds, of an outgoing request from the Oracle Identity Federation. The default is 120 seconds.

Encryption Settings

For Default XML Data Encryption Algorithm - Select one of the available encryption algorithms:

- AES-128 CBC
- Triple DES CBC
- AES-192 CBC
- AES-256 CBC

Note: Encryption methods other than AES-128 CBC require installation of the JCE encryption package. See Section 8.3, "Setting up JCE Policy Files for Oracle WebLogic Server".

Logout Options

You can configure one of these logout options:

- Failure on Error

  During the Global Logout flow, if an error is encountered, Oracle Identity Federation will either abort the operation and throw an error or continue and finish the logout operation.

- Status Return

  Return: If enabled and if the logout operation is started at Oracle Identity Federation by accessing the `/fed/user/logout` URL, the server redirects the user to the returnURL once the logout operation is done, and appends to that URL a query parameter indicating the result of the logout operation.

The query parameter name is orafed_slostatus, and the possible values are:

– 0 for success

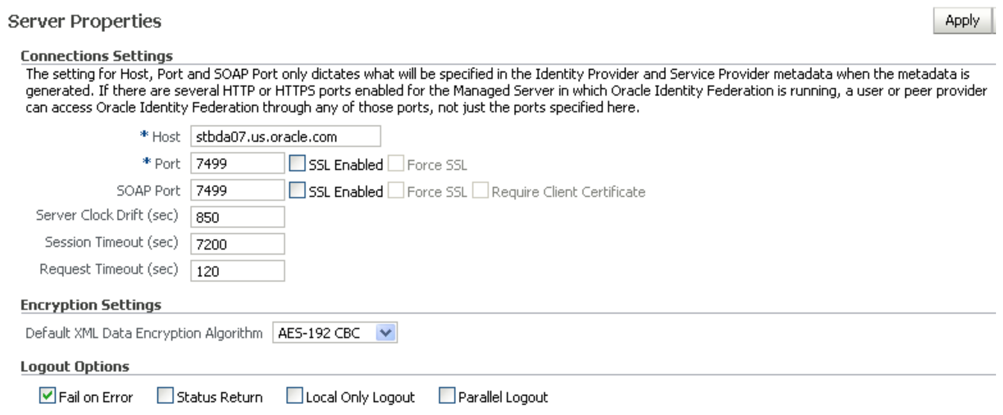
– 1 for failure

- Local Only Logout

  If enabled, when Oracle Identity Federation performs a logout operation for a user, it will not invoke the WS-Fed/SAML Logout protocol. Instead it only logs the user out of the Authentication Engines and Identity and Access Management framework, and destroys the Oracle Identity Federation session.

- Parallel Logout

  By default, when performing a SAML/WS-Fed Global Logout operation, Oracle Identity Federation sequentially redirects the user to all the providers from which the user needs to be logged out. This can become a time-consuming operation and if the logout flow is broken at one point because a provider is unresponsive, the logout flow will not finish. By enabling Parallel Logout, Oracle Identity Federation will display to the user a page with frames, each one performing a SAML/WS-Fed Logout operation with one specific provider. This can improve the global performance of the logout flow and minimize disruptions.



Connections Settings

The setting for Host, Port and SOAP Port only dictates what will be specified in the Identity Provider and Service Provider metadata when the metadata is generated. If there are several HTTP or HTTPS ports enabled for the Managed Server in which Oracle Identity Federation is running, a user or peer provider can access Oracle Identity Federation through any of those ports, not just the ports specified here.

* Host	stbda07.us.oracle.com
* Port	7499 ☐ SSL Enabled ☐ Force SSL
SOAP Port	7499 ☐ SSL Enabled ☐ Force SSL ☐ Require Client Certificate
Server Clock Drift (sec)	850
Session Timeout (sec)	7200
Request Timeout (sec)	120

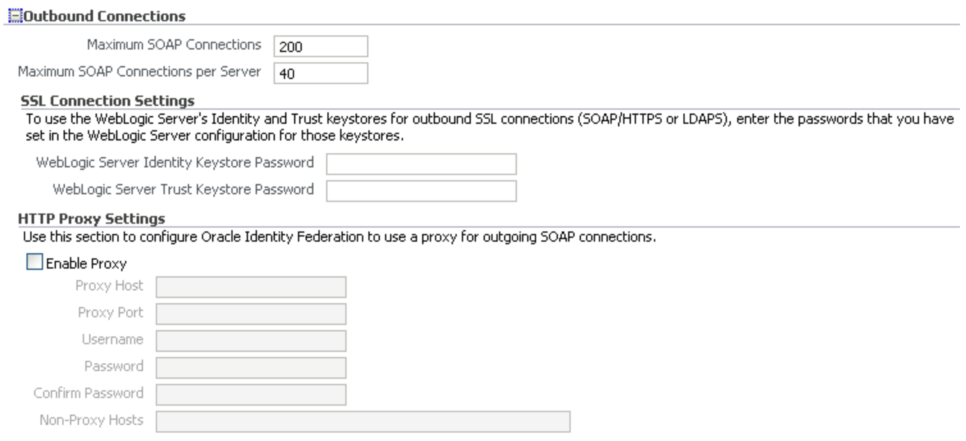
Encryption Settings

Default XML Data Encryption Algorithm: AES-192 CBC

Logout Options

☑ Fail on Error ☐ Status Return ☐ Local Only Logout ☐ Parallel Logout

5.2.2 Outbound Connection Properties



Outbound Connections

Maximum SOAP Connections	200
Maximum SOAP Connections per Server	40

SSL Connection Settings

To use the WebLogic Server's Identity and Trust keystores for outbound SSL connections (SOAP/HTTPS or LDAPS), enter the passwords that you have set in the WebLogic Server configuration for those keystores.

WebLogic Server Identity Keystore Password	
WebLogic Server Trust Keystore Password	

HTTP Proxy Settings

Use this section to configure Oracle Identity Federation to use a proxy for outgoing SOAP connections.

☐ Enable Proxy

Proxy Host	
Proxy Port	
Username	
Password	
Confirm Password	
Non-Proxy Hosts	

Outbound SOAP Connections

You can configure the following parameters:

- Maximum SOAP Connection
- Maximum SOAP Connection Per Server

Oracle Identity Federation can communicate with remote SAML Servers using different bindings, among them the SOAP binding. When Oracle Identity Federation needs to send a message to a remote server using the SOAP protocol, it will directly open a connection and send a SOAP message.

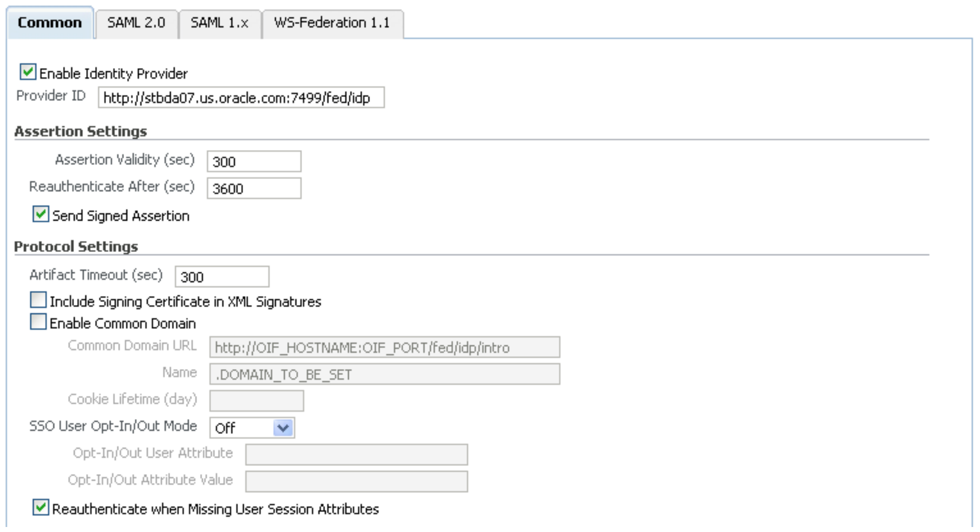
You can configure the maximum number of concurrent connections that Oracle Identity Federation can open when sending SOAP messages, and the maximum number of concurrent connections that Oracle Identity Federation can open when sending SOAP messages to a specific provider.

HTTP Proxy Settings

Use this section to configure Oracle Identity Federation to use a proxy for outgoing SOAP connections:

- Proxy Host - The proxy hostname.
- Proxy Port - The proxy port number.
- Username - The username to use when connecting to the proxy.
- Password - The password to use when connecting to the proxy.
- Non-Proxy Hosts - A list of hosts for which the proxy should not be used. Use ';' to separate multiple hosts.

5.3 Configuring Identity Providers - Common Properties



[✓] Enable Identity Provider
Provider ID: http://stbda07.us.oracle.com:7499/fed/idp

Assertion Settings

Assertion Validity (sec): 300
Reauthenticate After (sec): 3600
[✓] Send Signed Assertion

Protocol Settings

Artifact Timeout (sec): 300
[ ] Include Signing Certificate in XML Signatures
[ ] Enable Common Domain
Common Domain URL: http://OIF_HOSTNAME:OIF_PORT/fed/idp/intro
Name: .DOMAIN_TO_BE_SET
Cookie Lifetime (day):
SSO User Opt-In/Out Mode: Off
Opt-In/Out User Attribute:
Opt-In/Out Attribute Value:
[✓] Reauthenticate when Missing User Session Attributes

Enabling IdP

To enable the server as an IdP:

- Check the **Enable Identity Provider** box.
- Specify the Provider ID.

This is the URI for the Oracle Identity Federation instance. If it is a URL, it need not point to an actual resource.

Note: This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.

Assertion Settings

Specify assertion parameters as follows:

- Send Signed Assertion

  This determines whether the assertions issued by the identity provider will be signed.

- Assertion Validity

  This is the time, in seconds, during which an assertion issued by the identity provider is valid. An assertion is considered invalid if processed outside the validity period. The default is 300 seconds.

- Reauthenticate After

  This is the time, in seconds, after which the service provider must re-authenticate the user. Assertions containing an authentication statement by the identity provider are only valid for this period, after which the user is to be considered non-authenticated. The default is 3600 seconds.

Protocol Settings

Specify protocol settings as follows:

- Artifact Timeout

  This is the validity time, in seconds, of an artifact object created by Oracle Identity Federation. The default is 300 seconds.

- Include Signing Certificate in XML Signatures

  If checked, Oracle Identity Federation will add its signing certificate to the XML Digital Signature element of outgoing messages. This can be useful when the remote provider needs the signing certificate included in the message to be able to verify the signature created by Oracle Identity Federation.

- Common Domain

  See Also: Section 6.10, "Configuring the SAML 2.0 IdP Discovery (Common Domain Cookie) Profile"

  When an identity federation network contains multiple identity providers, a service provider needs to have a way to determine the identity provider(s) in use by a principal. This is achieved by utilizing a domain that is common to IdPs and SPs in the federation network, and sending to the user's browser a cookie, written in this domain, that lists all the IdPs where the user is logged in. Such a domain is known as a common domain, and the cookie identifying the IdPs is called a common domain cookie or introduction cookie.

  Check **Enable Common Domain** to specify that this IdP should set the introduction cookie. After every local authentication, Oracle Identity Federation redirects the user to the common domain, where the server can add its identifier to the introduction cookies at the user's browser.

Setting the common domain requires these parameters:

– Common Domain URL

  When an identity federation network contains multiple identity providers, a domain common to all providers is a way for a service provider to determine the identity provider(s) in use by a principal. After every authentication, a cookie on the user's browser (written in this domain) is updated with the IdPs identifier; the cookie lists all the user's IdPs and can be read by the service provider.

  Enter the URL where Oracle Identity Federation will read and set the IdP introduction cookie. The server listens on this URL, accepts requests, and updates the introduction cookie in the user's browser.

  Set this value only if you enabled the common domain.

– Name

  This is the common domain used for the IdP introduction cookie. It will be set as a cookie parameter on the introduction cookie. The value must begin with a dot (.) and must be of the form `.domain.suffix`. The default value is `.DOMAIN_TO_BE_SET`.

– Cookie Lifetime

  This is the lifetime, in days, of a common domain cookie issued by the IdP. If this field is set to 0 (default), the common domain cookie will be a session cookie.

- SSO User Opt-In/Opt-Out

  Determines if a user has given (or denied) permission to perform federated single sign-on for the user, based on the value of an attribute in the user's directory record.

  See Also: Section 6.18, "User Opt-In and Opt-Out for Single Sign-On" for details

- Reauthenticate when Missing User Session Attributes

  When Oracle Identity Federation acts as an IdP, it can use attributes stored in the session to populate the assertion. The session attributes are set:

  – during authentication, where a custom authentication engine provides attributes to be stored in the Oracle Identity Federation user session

  – when Oracle Identity Federation acts as a service provider. The content of the assertion (NameID, attributes...) can be saved in the user session; by default that data is not saved.

  When the assertion is created, Oracle Identity Federation/IdP will list the attributes it needs to retrieve from the user session and include in the assertion. If some attributes required in the assertion are missing from the user session, Oracle Identity Federation can be configured to either dismiss those attributes, or to invoke the authentication framework so that the custom authentication engine can provide those attributes to Oracle Identity Federation.

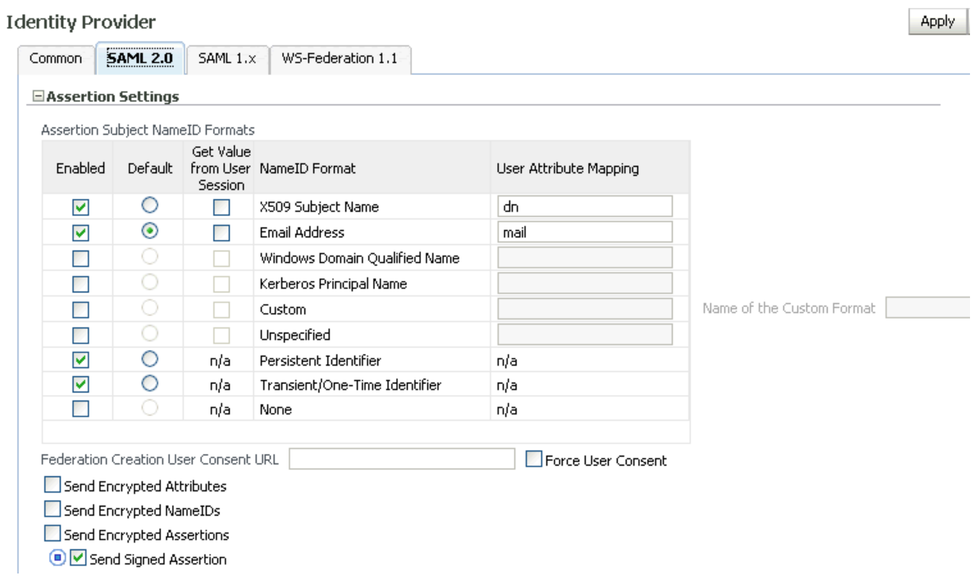
5.4 Configuring Identity Providers - Protocol-Specific Properties

This section describes how to configure IdP protocol-specific properties:

- Configure SAML 2.0 IdP Properties
- Configure SAML 1.x IdP Properties
- Configure WS-Federation IdP Properties
- Configure OpenID IdP Properties
- Configure an External OpenID Provider

5.4.1 Configure SAML 2.0 IdP Properties

Assertion Properties



Assertion Subject NameID Formats

Enabled	Default	Get Value from User Session	NameID Format	User Attribute Mapping
✓	○	☐	X509 Subject Name	dn
✓	●	☐	Email Address	mail
☐	○	☐	Windows Domain Qualified Name	
☐	○	☐	Kerberos Principal Name	
☐	○	☐	Custom	
☐	○	☐	Unspecified	
✓	○	n/a	Persistent Identifier	n/a
✓	○	n/a	Transient/One-Time Identifier	n/a
☐	○	n/a	None	n/a

Name of the Custom Format

Federation Creation User Consent URL
☐ Force User Consent
☐ Send Encrypted Attributes
☐ Send Encrypted NameIDs
☐ Send Encrypted Assertions
● ✓ Send Signed Assertion

Use this part of the page to maintain assertion name identifier formats for the SAML 2.0 protocol.

Provide the following information in the table:

- Enabled - Check a box to enable the desired format(s) that the Oracle Identity Federation instance will use as the SAML 2.0 name identifier value in IdP mode.
- Default - Use the radio button to select a default NameID format.
- Get Value from User Session - Check a box to specify that the attribute, to which the corresponding NameID maps, is found in the session created when the user is authenticated.
- NameID Format - This column displays the available name identifier formats. The formats are as follows:

Table 5–1 SAML 2.0 IdP NameID Formats

NameID Format	Default
X.509 Subject Name	dn
Email Address	mail
Windows Domain Qualified Name	empty
Kerberos Principal Name	empty

Table 5–1 (Cont.) SAML 2.0 IdP NameID Formats

NameID Format	Default
Custom	
Unspecified	
Persistent Identifier	
Transient/One-Time Identifier	
None	

- User Attribute Mapping - Enter the attribute name for the selected name ID format. Oracle Identity Federation uses the attribute name to perform a lookup in the user data store (or user session if **Get Value from User Session** is checked) for a name ID in this format.

Note: You can set the user attribute in the Assertion Subject NameID Formats table to "orafed-userid" to use the UserID to populate the NameID element. This is specially useful when no user store is configured for the IdP and thus no user store attributes are available.

Note: This table is used when creating an outgoing assertion. Depending on the NameID format, Oracle Identity Federation/IdP does the following:

- for X.509 Subject Name, Email Address, Windows Domain Qualified Name, Kerberos Principal Name and Unspecified, the server checks the configuration to determine which user attribute to use to populate the NameID element and retrieves the value from the user record.
- for Custom, the server checks the configuration to determine which user attribute to use to populate the NameID element and retrieves the value from the user record. It also sets the format of the NameID to the value specified in the configuration.
- for Persistent, the server uses a randomly generated identifier to build the NameID. That identifier is stored as a federation record for subsequent operations involving this user with the remote provider.
- for Transient, the server uses a randomly generated identifier to build the NameID that will only be used once.
- for None, the server does not include any NameID in the assertion.

Name of Custom Format - This is the format to use in the NameID, when creating an assertion that will use the custom NameID format.

UserID as the NameID

You can set the user attribute in the "Assertion Subject NameID Formats" table to "orafed-userid" to use the UserID to populate the NameID element. This feature is particularly useful when there is no user store configured for the IdP, and thus, no user store attributes available.

Additional Assertion Fields

The fields below the table are as follows:

- Federation Creation User Consent URL

  If the user must consent to setting up a new federation, this is the URL to which the user is redirected. You must design a consent page for this purpose.

- Force User Consent

  Check this box to force consent for setting up a new federation. If this box is checked, a user who is redirected to the federation server will explicitly have to accept or deny account linking in order to proceed.

- Send Encrypted Attributes

  Check this box to enable Oracle Identity Federation to send encrypted attributes to peer providers.

- Send Encrypted NameIDs

  Check this box to enable Oracle Identity Federation to send encrypted name identifiers to peer providers.

- Send Encrypted Assertions

  Check this box to enable Oracle Identity Federation to send encrypted assertions to peer providers.

- Send Signed Assertions

  Check this box to enable Oracle Identity Federation to send signed assertions to peer providers. The value specified on this page will override the value specified in the **Identity Provider - Common** tab, when using the SAML 2.0 protocol. If you do not wish to override the value in **Identity Provider - Common**, click the blue circle so that the square is not filled in, and there is an arrow pointing to the square, as in the image above.

About the User Consent Page

If the user must consent to setting up a new federation, you must design a consent page to which the user is redirected.

The server passes a number of query parameters to this URL:

Table 5–2 Parameters Passed to User Consent URL (SP Global)

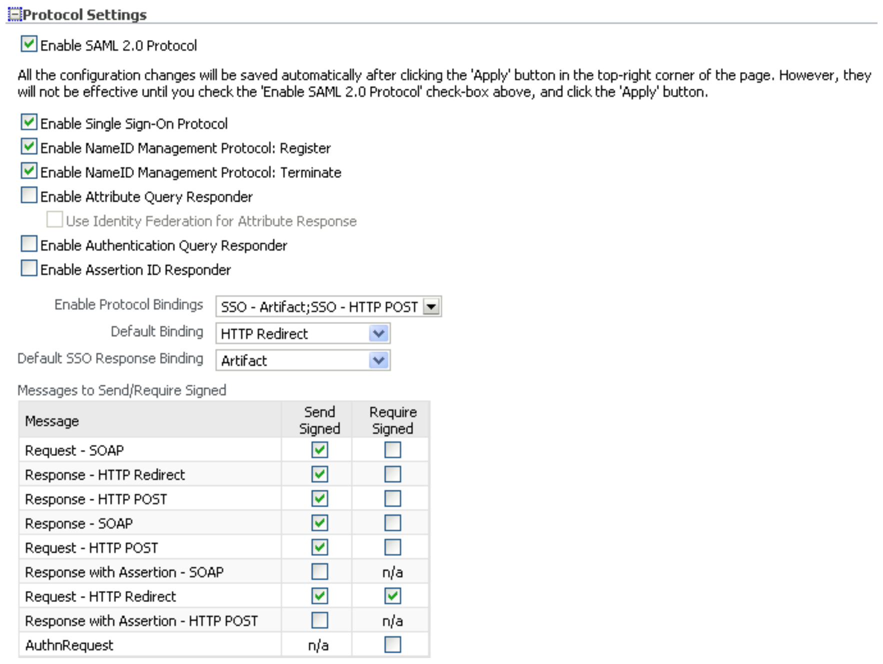
Parameter	Description
providerid	The peer provider id.
description	The description of the peer provider id.
returnurl	The URL to which the user should be directed once a consent decision has been made.
refid	Passed as a query parameter to the returnurl. Oracle Identity Federation requires this parameter in order to resume the operation the server had been performing prior to redirection to the consent URL.

When the consent URL page directs the user back to the return URL (by way of a link, form submission, or other means) it must pass two query parameters: the `refid` parameter described in the table, and a consent parameter indicating if consent was granted by the user (values are true or false).

Here is an example of a consent page:

```
<%
    String prefix = request.getContextPath();
    String redirectURL = request.getParameter("returnurl");
    String refID = request.getParameter("refid");
    String providerID = request.getParameter("providerid");
    String desc = request.getParameter("description");
%>
<HTML>
<BODY>
Do you consent to create a federation with <%=providerID%> (<%=desc%>):<br>
<form method="POST" action="<%=redirectURL%>">
    <input type="checkbox" name="userconsent" value="true"/>I agree<br>
    <input type="submit" value="OK" />
    <input type="hidden" name="refid"  value="<%=refID%>"/>
</form>
</BODY>
</HTML>
```

Protocol Settings



Protocol Settings

[✓] Enable SAML 2.0 Protocol

All the configuration changes will be saved automatically after clicking the 'Apply' button in the top-right corner of the page. However, they will not be effective until you check the 'Enable SAML 2.0 Protocol' check-box above, and click the 'Apply' button.

[✓] Enable Single Sign-On Protocol

[✓] Enable NameID Management Protocol: Register

[✓] Enable NameID Management Protocol: Terminate

[ ] Enable Attribute Query Responder

[ ] Use Identity Federation for Attribute Response

[ ] Enable Authentication Query Responder

[ ] Enable Assertion ID Responder

Enable Protocol Bindings	SSO - Artifact;SSO - HTTP POST
Default Binding	HTTP Redirect
Default SSO Response Binding	Artifact

Messages to Send/Require Signed

Message	Send Signed	Require Signed
Request - SOAP	✓	☐
Response - HTTP Redirect	✓	☐
Response - HTTP POST	✓	☐
Response - SOAP	✓	☐
Request - HTTP POST	✓	☐
Response with Assertion - SOAP	☐	n/a
Request - HTTP Redirect	✓	✓
Response with Assertion - HTTP POST	☐	n/a
AuthnRequest	n/a	☐

- Enable SAML 2.0 Protocol - Check this box to enable the SAML 2.0 protocol.
- Enable Single Sign-On Protocol - Check this box to enable the single sign-on protocol.
- Enable NameID Management Protocol: Terminate

  Check this box to enable the federation termination capability.

See Also: Section 1.2.4.8, "Federation Termination Profile"

Note: This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.

- Enable NameID Management Protocol: Register - Check this box to enable name ID registration.

  See Also: Section 1.2.4.5, "Name Identifier Management Profiles"

- Enable Attribute Query Responder

  Check this box to enable the identity provider to act as an attribute authority.

  Note: This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.

- Use Identity Federation for Attribute Response

  Check this box if you wish the user in the attribute request to be located in this identity provider using its federated identity. Note that if using this setting, the user must have a federation identity and its Name ID value and format must match the subject value and format specified in the `AttributeQuery`.

  When this box is checked, the attribute authority first tries to look up the user in the federation store; if no records are found, it locates the user by attribute value from the user data store.

  This property may also be overridden on the Edit Trusted Provider page on the Oracle Identity Federation Settings tab.

  See Also: Section 5.6.6, "Configuring Oracle Identity Federation as an IdP Attribute Responder"

- Enable Authentication Query Responder

  In SAML protocols, an identity provider may act as an authentication authority. A service provider may send an authentication query to an Authentication authority to ask "What assertions used for authentication have been issued for this subject"? The Authentication authority responds by providing the assertions that have been previously issued for authentication of the given subject.

  Check this box to enable the identity provider to act as an Authentication authority.

  Note: This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.

- Enable Assertion ID Responder

  In SAML protocols, an identity provider may act as an assertion ID authority. A trusted service provider may send an assertion ID request to an assertion ID authority in which it provides the unique ID of an assertion previously issued by the identity provider. The assertion ID authority responds by providing the assertion with the ID in the request.

Check this box to enable this identity provider to act as an assertion ID authority.

> **Note:** This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.

- Enable Protocol Bindings

  In the drop down, select all protocol bindings you wish to enable.
- Default Binding

  Select the binding to be used as a default when the identity provider sends a request or response (excluding SSO Responses) and no preferred binding was specified. e.g. in Name ID Management Protocol messages.
- Default SSO Response Binding

  Select the binding to be used as a default when the identity provider sends an SSO Response and no preferred binding was specified in the AuthnRequest.
- Messages to Send/Require Signed

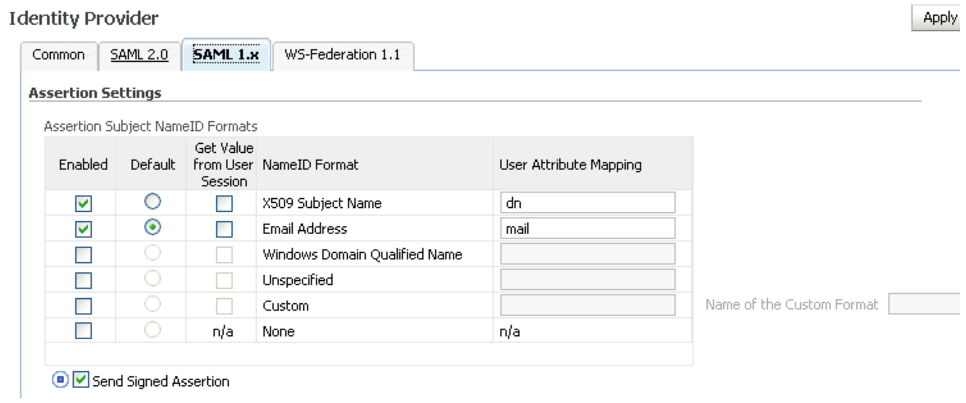
  specify the messages that Oracle Identity Federation sends, in IdP mode, that it must sign; and the messages it receives, in IdP mode, that it requires signed.

> **Note:** The Require AuthnRequest Signed property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.

5.4.2 Configure SAML 1.x IdP Properties

Use this page to configure Oracle Identity Federation to use the SAML 1.0/SAML 1.1 protocol when acting as an identity provider. The page contains tables for assertion settings and protocol settings, respectively.

Assertion Settings



Assertion Settings

Assertion Subject NameID Formats

Enabled	Default	Get Value from User Session	NameID Format	User Attribute Mapping
✓	○	☐	X509 Subject Name	dn
✓	●	☐	Email Address	mail
☐	○	☐	Windows Domain Qualified Name	
☐	○	☐	Unspecified	
☐	○	☐	Custom	
☐	○	n/a	None	n/a

Name of the Custom Format

✓ Send Signed Assertion

The table shows the Subject NameID formats you can configure. Provide the following information:

- Enabled - Check a box to enable the corresponding NameID format.

- Default - Use the radio button to select a default NameID format.
- Get Value from User Session - Check a box to specify that the attribute to which the corresponding NameID maps is found in the session created when the user is authenticated.
- NameID Format - This column displays the available name identifier formats. The formats are as follows:

Table 5–3 SAML 1.x Identity Provider Name ID Formats

NameID Format	Default
X.509 Subject Name	dn
Email Address	mail
Windows Domain Qualified Name	empty
Unspecified	empty
Custom	empty
None	

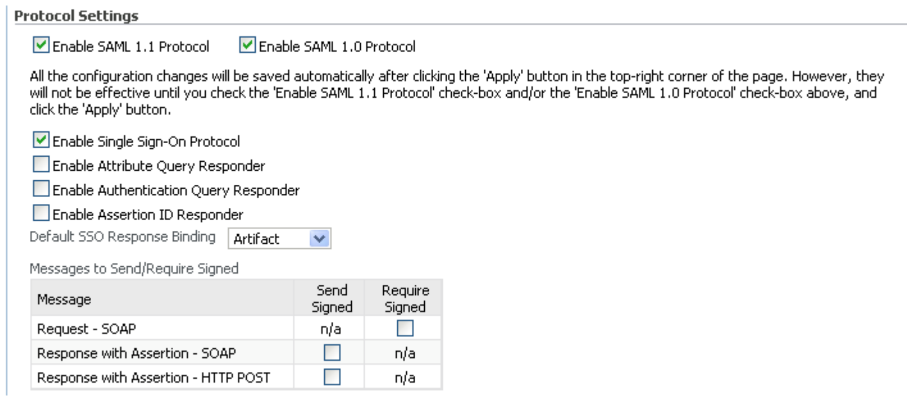
- User Attribute Mapping - Enter the attribute name for the selected name ID format. Oracle Identity Federation will use the attribute name to perform a lookup in the user data store (or user session if **Get Value from User Session** is checked) for a name ID in this format.

> **Note:** You can set the user attribute in the Assertion Subject NameID Formats table to "orafed-userid" to use the UserID to populate the NameID element. This is specially useful when no user store is configured for the IdP and thus no user store attributes are available.

Send Signed Assertion - Check this box to enable Oracle Identity Federation to send signed assertions to peer providers. The value specified on this page will override the value specified in the **Identity Provider - Common** tab, when using the SAML 1.0 or SAML 1.1 protocols. If you do not wish to override the value in **Identity Provider - Common**, click the blue circle so that the square is not filled in, and there is an arrow pointing to the square, as in the image above.

Protocol Settings

Use this table to specify protocol settings and related attributes.



Protocol Settings

- [✓] Enable SAML 1.1 Protocol
- [✓] Enable SAML 1.0 Protocol

All the configuration changes will be saved automatically after clicking the 'Apply' button in the top-right corner of the page. However, they will not be effective until you check the 'Enable SAML 1.1 Protocol' check-box and/or the 'Enable SAML 1.0 Protocol' check-box above, and click the 'Apply' button.

- [✓] Enable Single Sign-On Protocol
- [ ] Enable Attribute Query Responder
- [ ] Enable Authentication Query Responder
- [ ] Enable Assertion ID Responder

Default SSO Response Binding: Artifact

Messages to Send/Require Signed

Message	Send Signed	Require Signed
Request - SOAP	n/a	☐
Response with Assertion - SOAP	☐	n/a
Response with Assertion - HTTP POST	☐	n/a

Provide the following information:

- Protocol - Check one or both of the Enable SAML 1.1 Protocol and Enable SAML 1.0 Protocol boxes.
- Enable Single Sign-On - Check this box to enable the single sign-on protocol.
- Enable Attribute Query Responder - Check this box to enable the identity provider to act as an attribute authority.
- Enable Authentication Query Responder

  In SAML protocols, an identity provider may act as an Authentication authority. A service provider may send an Authentication query to an Authentication authority to ask "What assertions used for authentication have been issued for this subject"? The Authentication authority responds by providing the assertions that have been previously issued for authentication of the given subject.

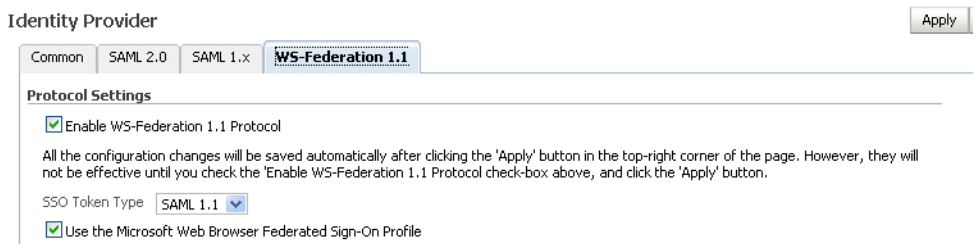
  Check this box to enable the identity provider to act as an Authentication authority.
- Enable Assertion ID Responder

  In SAML protocols, an identity provider may act as an assertion ID authority. A trusted service provider may send an assertion ID request to an assertion ID authority in which it provides the unique ID of an assertion previously issued by the identity provider. The assertion ID authority responds by providing the assertion with the ID in the request.

  Check this box to enable this identity provider to act as an assertion ID authority.
- Default SSO Response Binding - Select the binding to be used as a default when the identity provider sends an SSO Response and no preferred binding was specified in the AuthnRequest.
- Messages to Send/Require Signed - specify the messages that Oracle Identity Federation sends, in IdP mode, that it must sign; and the messages it receives, in IdP mode, that it requires signed.

5.4.3 Configure WS-Federation IdP Properties

Use this page to configure Oracle Identity Federation to use the WS-Federation 1.1 protocol when acting as an identity provider.



✓ Enable WS-Federation 1.1 Protocol

All the configuration changes will be saved automatically after clicking the 'Apply' button in the top-right corner of the page. However, they will not be effective until you check the 'Enable WS-Federation 1.1 Protocol check-box above, and click the 'Apply' button.

SSO Token Type: SAML 1.1

✓ Use the Microsoft Web Browser Federated Sign-On Profile

Provide the following information:

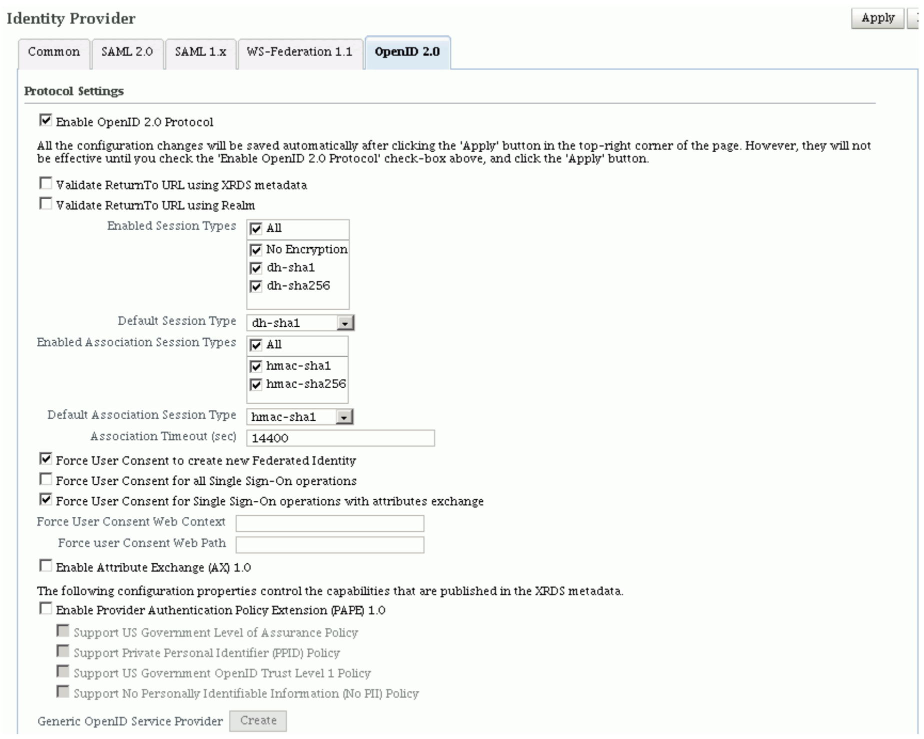
- Enable WS-Federation 1.1 Protocol - Check this box to enable the protocol.
- SSO Token Type - Use the drop-down box to select the single sign-on token type.
- Use Microsoft Web Browser Federated SSO Profile - Check this box to have Oracle Identity Federation use the Microsoft WS-Fed protocol specifications.

5.4.4 Configure OpenID IdP Properties

Use this page to configure an OpenID provider in IdP mode.

See Also: .

- Section 2.2.1.3, "OpenID 2.0 Protocol"
- Section 5.4.5, "Configure an External OpenID Provider"


Identity Provider

Apply

Common | SAML 2.0 | SAML 1.x | WS-Federation 1.1 | **OpenID 2.0**

Protocol Settings

[✓] Enable OpenID 2.0 Protocol

All the configuration changes will be saved automatically after clicking the 'Apply' button in the top-right corner of the page. However, they will not be effective until you check the 'Enable OpenID 2.0 Protocol' check-box above, and click the 'Apply' button.

[ ] Validate ReturnTo URL using XRDS metadata
[ ] Validate ReturnTo URL using Realm

Enabled Session Types: [✓] All, [✓] No Encryption, [✓] dh-sha1, [✓] dh-sha256

Default Session Type: dh-sha1

Enabled Association Session Types: [✓] All, [✓] hmac-sha1, [✓] hmac-sha256

Default Association Session Type: hmac-sha1

Association Timeout (sec): 14400

[✓] Force User Consent to create new Federated Identity
[ ] Force User Consent for all Single Sign-On operations
[✓] Force User Consent for Single Sign-On operations with attributes exchange

Force User Consent Web Context:

Force user Consent Web Path:

[ ] Enable Attribute Exchange (AX) 1.0

The following configuration properties control the capabilities that are published in the XRDS metadata.

[ ] Enable Provider Authentication Policy Extension (PAPE) 1.0
[ ] Support US Government Level of Assurance Policy
[ ] Support Private Personal Identifier (PPID) Policy
[ ] Support US Government OpenID Trust Level 1 Policy
[ ] Support No Personally Identifiable Information (No PII) Policy

Generic OpenID Service Provider: Create

The fields are as follows:

- Enable OpenID 2.0 Protocol - Check this box to enable the OpenID 2.0 protocol.
- Validate ReturnTo URL using XRDS metadata - Check this box to verify the ReturnTo URL (which is the URL to which the server redirects once the authentication is processed) using the XRDS.
- Validate ReturnTo URL using Realm - Check this box to specify that the IdP should validate the ReturnTo URL present in the OpenID authentication request using the realm.
- Enabled Session Types - If validating ReturnURL using realm, check the boxes to enable specific session types or `All` to enable all session types.
- Default Session Type - If validating ReturnURL using realm, use the drop-down to select the default session type. Any of the enabled session types may be selected.
- Enable Association Session Types - Lists the enabled association types that the OpenID provider supports:
  – No encryption
  – Diffie-Hellman SHA1
  – Diffie-Hellman SHA256
- Default Association Session Type - Specifies the default association session type.

Note: Changes in session type and/or association session type require restart of the Oracle Identity Federation server.

- Association Timeout (sec) -This is the duration of validity of the association in seconds.
- Force User Consent to create new Federated Identity - Check this box to force consent for setting up a new federation.
- Force User Consent for all Single Sign-On Operations - Check this box to force consent for SSO.
- Force User Consent for Single Sign-On Operations with attributes exchange - Check this box to prompt the user for consent any time an SSO operation is being performed between Oracle Identity Federation/IdP and the RP, where user attributes will be released to the RP.

  Related fields for this feature:

  – Force User Consent Web Context - Contains the Web Context of the OpenID consent page to be used instead of the Oracle Identity Federation OpenID built-in consent page. If filled, it will reference a user-implemented page on Oracle WebLogic Server.

  – Force User Consent Web Path
- Enable Attribute Exchange (AX) 1.0 - Check this box to enable the AX extension.
- Enable PAPE 1.0 - Check this box to enable the PAPE 1.0 extension. With this feature you can specify one or more of:

  – US Government Level of Assurance Policy

  – PPID Policy

  – US Government OpenID Trust Level 1 Policy

  – US Government No PII Policy
- Generic OpenID Service Provider

  Oracle Identity Federation lets you define the OpenID RP as:

  – a federated partner for which you can define specific settings

  – an unknown RP with general attribute mappings

  Click **Create** to define a generic OpenID RP.

5.4.5 Configure an External OpenID Provider

Section 5.4.4 describes how to enable the out-of-the-box Oracle Identity Federation OpenID provider.

You can also configure an external OpenID provider so that Oracle Identity Federation acts as the relying party (RP/SP) and an external resource acts as the OpenID provider (OP). Google and Yahoo are examples of external OpenID providers.

Take the following steps to configure an external OpenID provider:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control.
2. Navigate to the Oracle Identity Federation instance.
3. Select **Administration**, then **Federations**.

4. Click **Add** to add a new OpenID provider.
5. In the pop-up box, select "Add provider manually".
6. Enter the provider ID using a URL in this format:

   ```
   http://node123.us.example.com:7777/fed/idp
   ```

7. For protocol version, select "OpenID2.0".
8. For provider type, select "Identity Provider".
9. Click **OK** to create the provider.
10. Edit the new provider. Enter the provider's discovery URL in this format:

    ```
    http://node123.us.example.com:7777/fed/idp
    ```

    or enter the provider's OpenID endpoint URL if the IdP does not support OpenID discovery.
11. Click **Apply** to commit the edits.

This procedure follows the basic steps for configuring a provider as shown in Section 4.3.2, "Add Trusted Providers".

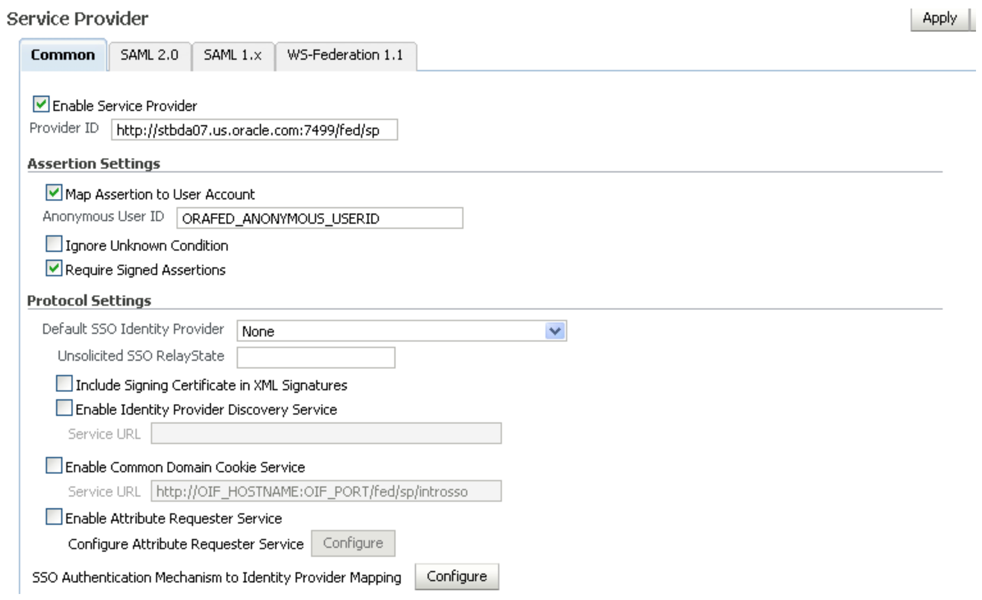
5.5 Configuring Service Providers

This section describes how to edit and update the protocol-specific service provider (SP) properties in Oracle Identity Federation. It contains these sub-sections:

- Configure Service Provider - Common Properties
- Configure SAML 2.0 SP Properties
- Configure SAML 1.x SP Properties
- Configure WS-Federation 1.1 SP Properties
- Configure OpenID SP Properties

5.5.1 Configure Service Provider - Common Properties

Use this table to configure SP properties common to all protocols.



Service Provider

Common

☑ Enable Service Provider

Provider ID: http://stbda07.us.oracle.com:7499/fed/sp

Assertion Settings

☑ Map Assertion to User Account

Anonymous User ID: ORAFED_ANONYMOUS_USERID

☐ Ignore Unknown Condition

☑ Require Signed Assertions

Protocol Settings

Default SSO Identity Provider: None

Unsolicited SSO RelayState:

☐ Include Signing Certificate in XML Signatures

☐ Enable Identity Provider Discovery Service

Service URL:

☐ Enable Common Domain Cookie Service

Service URL: http://OIF_HOSTNAME:OIF_PORT/fed/sp/introsso

☐ Enable Attribute Requester Service

Configure Attribute Requester Service

SSO Authentication Mechanism to Identity Provider Mapping

Assertion Settings

- Enable Map Assertion to User Account - Check this box if you wish Oracle Identity Federation to map the assertion to a user account. Disable it if you wish to implement a custom SP Engine to do the mapping instead.
- Anonymous User ID - Enter the User ID that will be passed to the SP engine if the assertion received cannot be mapped to a user account, either because mapping assertions to user accounts is disabled or the format of the Name ID in the assertion is 'transient'.
- Ignore Unknown Condition - Check this box to have the service provider ignore any conditions it does not recognize in the assertion sent by the identity provider.
- Require Signed Assertions - Check this box to require assertions received from identity providers to be signed.

Protocol Settings

- Default SSO Identity Provider - Select the identity provider to which requests should be sent as a default when an SSO operation is initiated and no preferred identity provider is specified.
- Unsolicited SSO RelayState - When an Oracle Identity Federation SP receives an unsolicited assertion, it sends the user to the relay state specified by the assertion following the SSO operation; if the relay state field in the assertion is empty, it will use the Unsolicited SSO RelayState to redirect the user.
- Include Signing Certificate in XML Signatures - check this box to have Oracle Identity Federation include the signing certificate in the signature when signing XML messages.
- Enable Identity Provider Discovery Service

  Oracle Identity Federation provides a service, called the Identity Provider Discovery Service, in which the user can be redirected to a custom page in which he can select the identity provider from which he wishes to authenticate.

  Check this box to enable the Identity Provider Discovery Service, and enter the Service URL of the custom page where the user can select the identity provider to be used.

See Also: Section 6.11, "Configuring the Identity Provider Discovery Service"

- Enable Common Domain Cookie Service

  When an identity federation network contains multiple identity providers, a service provider needs to have a way to determine the identity provider(s) in use by a principal. This is achieved by utilizing a domain that is common to IdPs and SPs in the federation network, and sending to the user's browser a cookie, written in this domain, that lists all the IdPs where the user is logged in. Such a domain is known as a common domain, and the cookie identifying the IdPs is called a common domain cookie or introduction cookie.

  Check this box to specify that this SP should read the introduction cookie, and enter the Service URL where Oracle Identity Federation will read the introduction cookie.

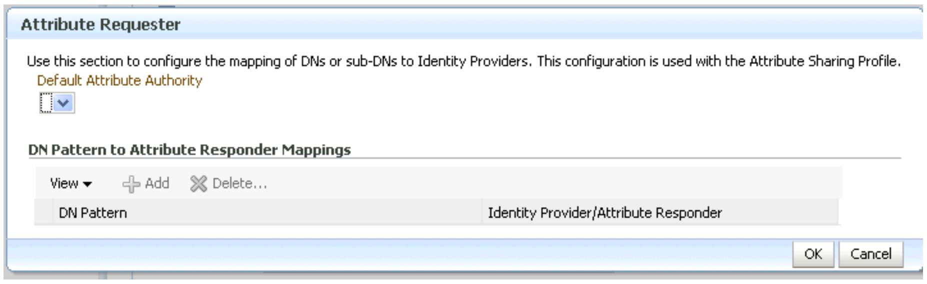
  See Also: Section 6.10, "Configuring the SAML 2.0 IdP Discovery (Common Domain Cookie) Profile"

- Enable Attribute Requester Service - check this box to enable this service provider to act as an Attribute Requester.

  See Also: Section 5.6.5, "Configuring Oracle Identity Federation as an SP Attribute Requester"

Note: This property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.

Configure Attribute Requester Service


Attribute Requester

Use this section to configure the mapping of DNs or sub-DNs to Identity Providers. This configuration is used with the Attribute Sharing Profile.

Default Attribute Authority

DN Pattern to Attribute Responder Mappings

DN Pattern	Identity Provider/Attribute Responder

- Default Attribute Authority - Select the attribute authority to which Attribute Queries should be sent to as a default, when no attribute authority is specified in the request.
- DN Pattern to Attribute Responder Mappings - Use this table to map User DN patterns to attribute authorities. When sending an attribute query for a given user, Oracle Identity Federation will look at the user's DN, match it to a pattern on this table, and send the attribute query to the corresponding attribute authority. If no pattern matches the user's DN, the default attribute authority is used.

  By default, the DN pattern is case-sensitive, and case is considered in comparing the user DN to the DN pattern. You can make the comparison case-insensitive by using the WLST configuration command:

```
setConfigProperty ('dnidpmapping','caseinsensitive','true','boolean')
```

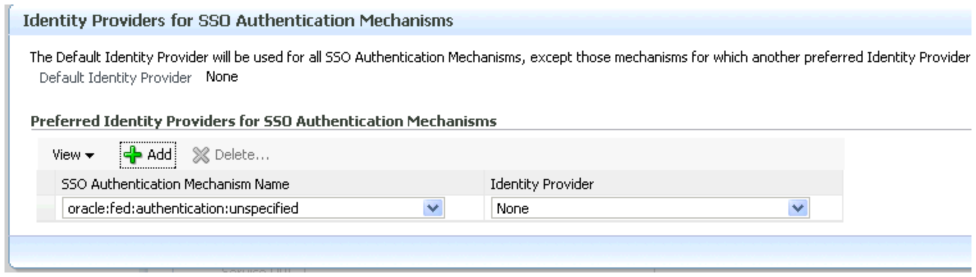
Use `true` for case-insensitive comparison, `false` for case-sensitive comparison.

> **See Also:** Section 5.6.5, "Configuring Oracle Identity Federation as an SP Attribute Requester"

Identity Providers for SSO Authentication Mechanism

Use this table to map authentication mechanisms to identity providers. When an SSO operation is initiated and no identity provider is specified, Oracle Identity Federation will look at this table to map the requested authentication mechanism to an identity provider and send the `AuthnRequest` to this identity provider. If the authentication mechanism cannot be mapped to an identity provider, the default SSO identity provider will be used.

> **See Also:** Section 5.14, "Configuring Authentication Mechanisms"


Identity Providers for SSO Authentication Mechanisms

The Default Identity Provider will be used for all SSO Authentication Mechanisms, except those mechanisms for which another preferred Identity Provider

Default Identity Provider None

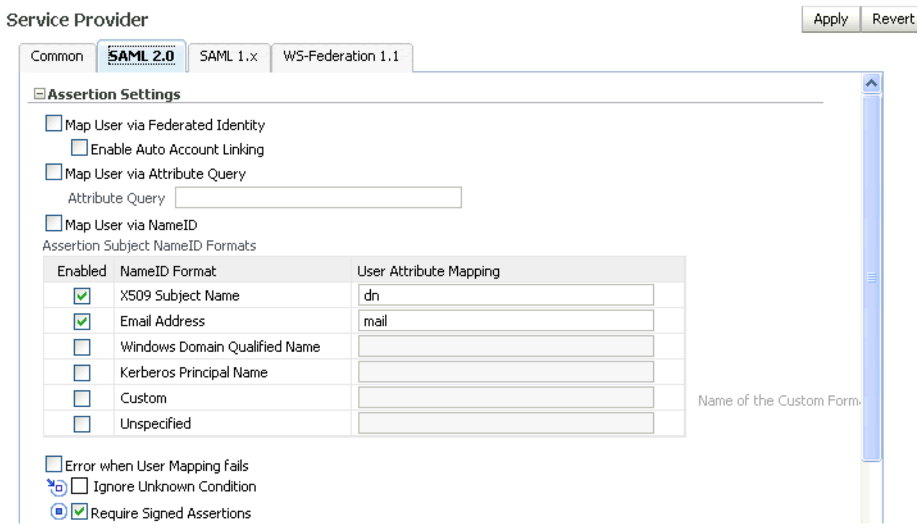
Preferred Identity Providers for SSO Authentication Mechanisms

SSO Authentication Mechanism Name	Identity Provider
oracle:fed:authentication:unspecified	None

5.5.2 Configure SAML 2.0 SP Properties

Use this tab to maintain Oracle Identity Federation properties in service provider mode under the SAML 2.0 protocol.

Assertion Settings


Service Provider

Assertion Settings

- [ ] Map User via Federated Identity
  - [ ] Enable Auto Account Linking
- [ ] Map User via Attribute Query

Attribute Query

- [ ] Map User via NameID

Assertion Subject NameID Formats

Enabled	NameID Format	User Attribute Mapping
✓	X509 Subject Name	dn
✓	Email Address	mail
	Windows Domain Qualified Name	
	Kerberos Principal Name	
	Custom	
	Unspecified	

Name of the Custom Form

- [ ] Error when User Mapping fails
- [ ] Ignore Unknown Condition
- [x] Require Signed Assertions

Choose one of these methods of assertion mapping by checking the associated box:

- Federated Identity
- Attribute Query

- Subject NameID

If mapping users through federated identity, check the box labeled Enable Auto Account Linking.

> **See Also:** Section 6.17, "Automatic Account Linking Based on Attribute Query Mapping"

If mapping users through attribute query, enter the query string in the associated box.

If mapping users through subject NameID, check the box and select the applicable NameID formats from the table titled Assertion Subject NameID Formats. Provide this information in the table:

- Check the corresponding **Enabled** box to enable the desired format(s) that the Oracle Identity Federation instance will support in SP mode.
- NameID Format - This column displays the available SAML 2.0 NameID formats.
- User Attribute Mapping - Enter the attribute name for the selected name ID format. Oracle Identity Federation will use this attribute name to perform a lookup in the user data store for a name ID in this format.

  The name identifier formats are as follows:

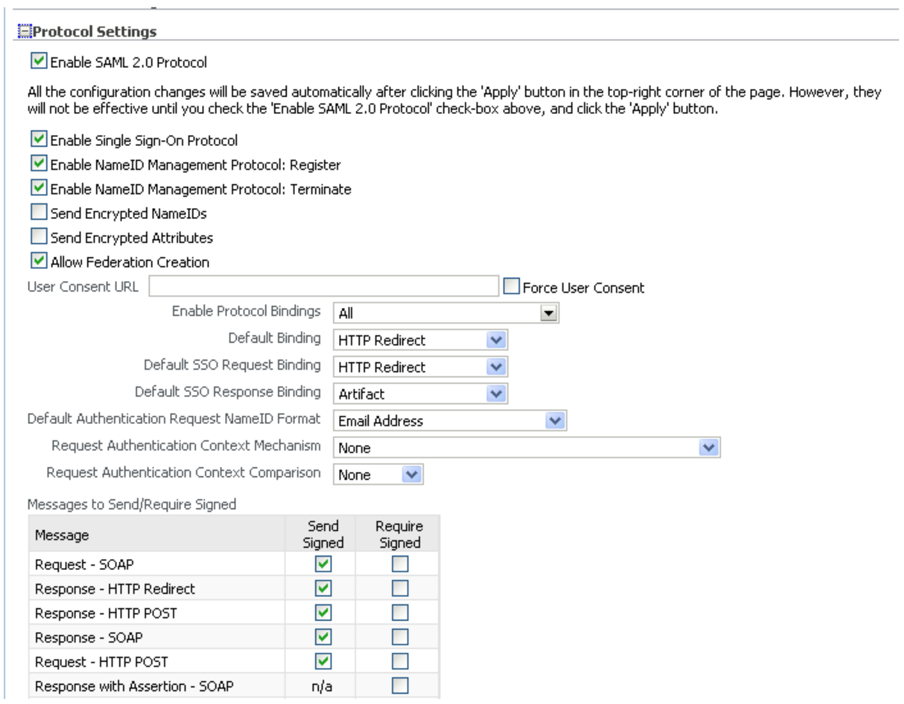
Table 5–4 SAML 2.0 SP Name ID Formats

NameID Format	Default
X.509 Subject Name	dn
Email Address	mail
Windows Domain Qualified Name	empty
Kerberos Principal Name	
Custom	empty
Unspecified	empty

Name of the Custom Format - When processing an assertion, this is the name of the format that will be mapped to the custom NameID format type.

Additionally, you can check **Error when User Mapping Fails** to indicate how Oracle Identity Federation should handle mapping errors.

Protocol Settings



Protocol Settings

☑ Enable SAML 2.0 Protocol

All the configuration changes will be saved automatically after clicking the 'Apply' button in the top-right corner of the page. However, they will not be effective until you check the 'Enable SAML 2.0 Protocol' check-box above, and click the 'Apply' button.

☑ Enable Single Sign-On Protocol
☑ Enable NameID Management Protocol: Register
☑ Enable NameID Management Protocol: Terminate
☐ Send Encrypted NameIDs
☐ Send Encrypted Attributes
☑ Allow Federation Creation

User Consent URL	☐ Force User Consent
Enable Protocol Bindings	All
Default Binding	HTTP Redirect
Default SSO Request Binding	HTTP Redirect
Default SSO Response Binding	Artifact
Default Authentication Request NameID Format	Email Address
Request Authentication Context Mechanism	None
Request Authentication Context Comparison	None

Messages to Send/Require Signed

Message	Send Signed	Require Signed
Request - SOAP	☑	☐
Response - HTTP Redirect	☑	☐
Response - HTTP POST	☑	☐
Response - SOAP	☑	☐
Request - HTTP POST	☑	☐
Response with Assertion - SOAP	n/a	☐

Provide the following information:

- Enable SAML 2.0 Protocol - Check the box to enable this protocol for the SP.
- Enable Single Sign-On Protocol - Check the box to enable the single sign-on protocol.
- Enable NameID Management Protocol: Register - Check the box to enable NameID registration.

  See Also: Section 1.2.4.5, "Name Identifier Management Profiles"

- Enable Federation Termination Protocol - Check this box to enable the federation termination protocol.

  See Section 1.2.4.8, "Federation Termination Profile" for an explanation of this feature.

- Send Encryption NameIDs - Check this box to enable Oracle Identity Federation to send encrypted name identifiers to peer providers.
- Send Encryption Attributes - Check this box to enable Oracle Identity Federation to send encrypted attributes to peer providers.
- Allow Federation Creation - Check this box to allow federation creation. This is required if you configure the SP to request persistent NameID format as described below.
- Force User Consent - Check this box to force consent for setting up a new federation. A user who is redirected to the federation server will explicitly have to accept or deny account linking in order to proceed.
- User Consent URL - Enter the URL to be displayed to the user to obtain consent for federation.

  The server passes a number of query parameters to this URL:

See Also: Section 5.4.1, "Configure SAML 2.0 IdP Properties" for an example showing how the query parameters are used.

Table 5–5 Parameters Passed to User Consent URL (Local Setting)

Parameter	Description
providerid	This is the peer provider id.
description	This is the description of the peer provider id.
returnurl	This is the URL to which the user should be directed once a consent decision has been made.
refid	This is passed as a query parameter to the returnurl. Oracle Identity Federation require this parameter in order to resume the operation the server had been performing prior to redirection to the consent URL.

- Enable Protocol Bindings - Specify the valid bindings using the drop-down list.
- Default Binding - Specifies the preferred binding to use, when possible, in sending messages to peer providers. Valid values are:
  - HTTP Redirect
  - HTTP POST
  - HTTP Post Simple Sign
  - SOAP
- Default SSO Request Binding - Specifies the preferred binding for the service provider to use, when possible, in sending authentication requests to the identity provider. Use only if this server instance is acting as a service provider. Valid values are:
  - HTTP Redirect
  - HTTP POST
  - HTTP Post Simple Sign
- Default SSO Response Binding - Specifies the preferred binding for the identity provider to use, when possible, in sending an assertion to the service provider. Valid values are:
  - Artifact
  - HTTP POST
  - HTTP POST Simple Sign
- Default Authentication Request NameID Format - Use the list box to select a default name ID format for authentication requests. Choices are:
  - X.509 Subject Name
  - Email Address
  - Windows Domain Qualified Name
  - Kerberos Principal Name
  - Persistent identifier
  - Transient/one-time identifier

    - Unspecified

      If the default authentication request NameID format at the SP is unspecified, the IdP will use the default assertion NameID format when creating the assertion (for example, email NameID format).

- Request Authentication Context Mechanism - Use this list box to select the authentication mechanism that this service provider will specify in the AuthnRequest to the identity provider.

  See Also: Section 5.14, "Configuring Authentication Mechanisms"

- Request Authentication Context Comparison - Use the list box to select the authentication context comparison that this service provider will specify in the AuthnRequest sent to the identity provider.
- Messages to Send/Require Signed

  Use this table when configuring a service provider to specify which message types that provider should send signed and/or require signed.

  Notes:

  - The Require Signed Assertion property affects server metadata. When updating this property, distribute the updated metadata to all trusted providers.
  - If you configure the SP to request Persistent NameID format (or if it expects to receive Persistent from the IdP in case the SP does not specify a format), then the SP has to be configured to allow federation creation.
  - Although your configuration changes are saved when you click **Apply**, at least one of the protocol boxes must also be checked to ensure that the changes on this page are effective.

5.5.3 Configure SAML 1.x SP Properties

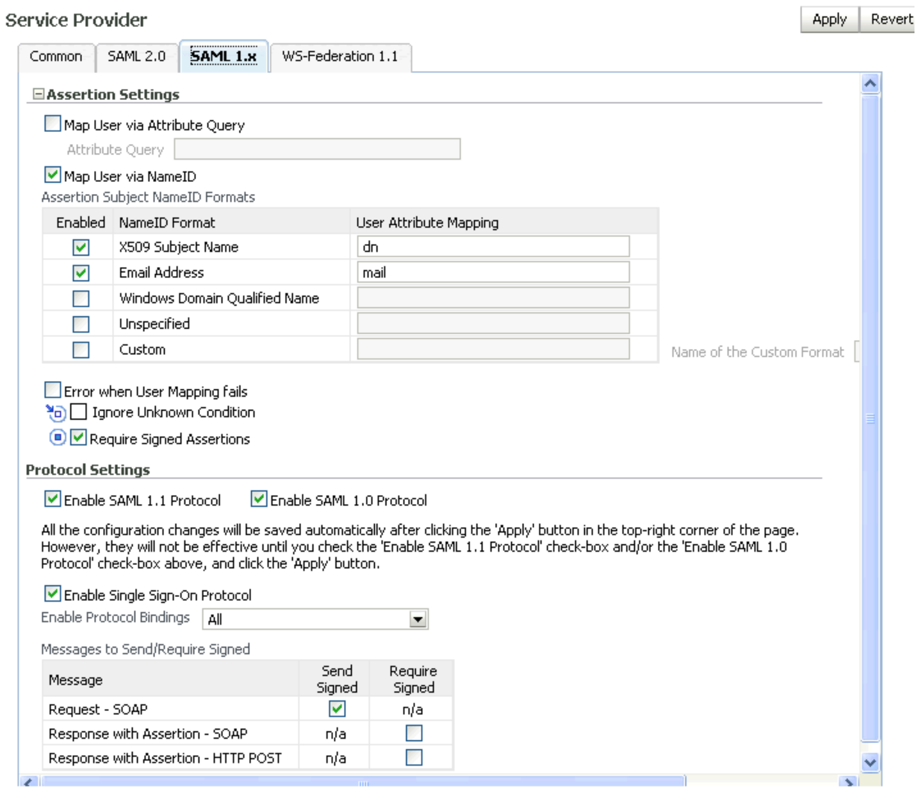
Use this tab to specify configuration details for Oracle Identity Federation SAML 1.x domains.

Assertion Settings

Select one of these mapping choices:

- Map User via Attribute Query - Check the box and enter an attribute query.
- Map User via NameID - Check the box and select the applicable NameID formats from the table titled Assertion Subject NameID Formats.

Additionally, you can check **Error when User Mapping Fails** to indicate how Oracle Identity Federation should handle mapping errors.



Assertion Settings

☐ Map User via Attribute Query

Attribute Query

☑ Map User via NameID

Assertion Subject NameID Formats

Enabled	NameID Format	User Attribute Mapping
✓	X509 Subject Name	dn
✓	Email Address	mail
☐	Windows Domain Qualified Name	
☐	Unspecified	
☐	Custom	

Name of the Custom Format

☐ Error when User Mapping fails

☐ Ignore Unknown Condition

☑ Require Signed Assertions

Protocol Settings

☑ Enable SAML 1.1 Protocol ☑ Enable SAML 1.0 Protocol

All the configuration changes will be saved automatically after clicking the 'Apply' button in the top-right corner of the page. However, they will not be effective until you check the 'Enable SAML 1.1 Protocol' check-box and/or the 'Enable SAML 1.0 Protocol' check-box above, and click the 'Apply' button.

☑ Enable Single Sign-On Protocol

Enable Protocol Bindings: All

Messages to Send/Require Signed

Message	Send Signed	Require Signed
Request - SOAP	✓	n/a
Response with Assertion - SOAP	n/a	☐
Response with Assertion - HTTP POST	n/a	☐

To Use the Table of Assertion Subject NameID Formats

If you selected assertion mapping through subject NameIDs, provide this information in the table:

- Check the corresponding **Enabled** box to enable the desired format(s) that the Oracle Identity Federation instance will support as SAML 1.1/1.0 name identifier formats in SP mode.
- NameID Format - This column displays the available SAML 1.x NameID formats.
- User Attribute Mapping - Enter the attribute name for the selected name ID format. Oracle Identity Federation will use this attribute name to perform a lookup in the user data store for a name ID in this format.

  The name identifier formats are as follows:

Table 5–6 SAML 1.1/1.0 SP Name ID Formats

NameID Format	Default
X.509 Subject Name	dn
Email Address	mail
Windows Domain Qualified Name	empty
Unspecified	empty
Custom	empty

Protocol Settings

Provide the following information:

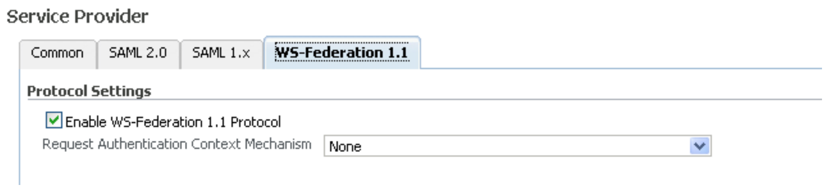
- Enable SAML 1.1 Protocol - Check the box to enable this protocol for the SP.

- Enable SAML 1.0 Protocol - Check the box to enable this protocol for the SP.
- Enable Protocol Bindings - Use the drop-down to select the binding to use.
- Messages to Send/Require Signed - Use this table when configuring a service provider to specify which message types that provider should send signed

Note: Although your configuration changes are saved when you click **Apply**, at least one of the protocol boxes must also be checked to ensure that the changes on this page are effective.

5.5.4 Configure WS-Federation 1.1 SP Properties

Use this page to configure Oracle Identity Federation to use the WS-Federation 1.1 protocol when acting as a service provider.


Service Provider

Common | SAML 2.0 | SAML 1.x | WS-Federation 1.1

Protocol Settings

☑ Enable WS-Federation 1.1 Protocol

Request Authentication Context Mechanism: None

Provide the following information:

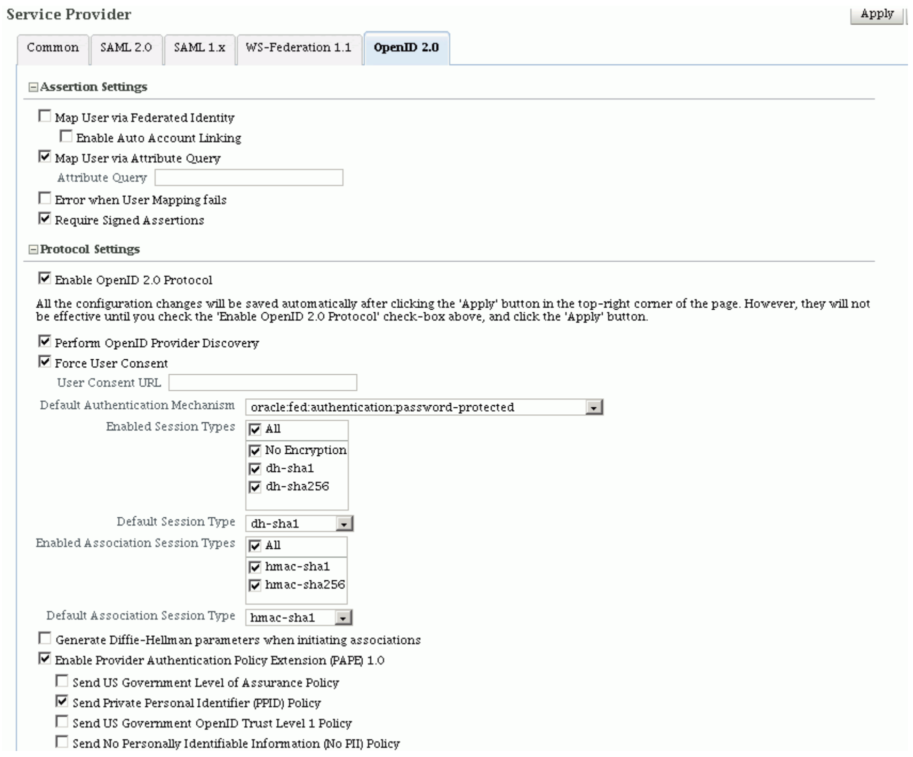
- Enable WS-Federation 1.1 Protocol - Check the box to enable WS-Federation 1.1 protocol for the provider.
- Request Authentication Context Mechanism - Use the drop-down to select the authentication mechanism that will be sent in the authentication request to the identity provider.

Click **Apply** to save your changes, or **Revert** to restore the screen to its previous values.

5.5.5 Configure OpenID SP Properties

Use this page to configure the OpenID protocol in SP mode.

See Also: Section 2.2.1.3, "OpenID 2.0 Protocol".



Service Provider

Apply

Common | SAML 2.0 | SAML 1.x | WS-Federation 1.1 | **OpenID 2.0**

⊟ **Assertion Settings**

☐ Map User via Federated Identity

☐ Enable Auto Account Linking

☑ Map User via Attribute Query

Attribute Query

☐ Error when User Mapping fails

☑ Require Signed Assertions

⊟ **Protocol Settings**

☑ Enable OpenID 2.0 Protocol

All the configuration changes will be saved automatically after clicking the 'Apply' button in the top-right corner of the page. However, they will not be effective until you check the 'Enable OpenID 2.0 Protocol' check-box above, and click the 'Apply' button.

☑ Perform OpenID Provider Discovery

☑ Force User Consent

User Consent URL

Default Authentication Mechanism: oracle:fed:authentication:password-protected

Enabled Session Types: ☑ All; ☑ No Encryption; ☑ dh-sha1; ☑ dh-sha256

Default Session Type: dh-sha1

Enabled Association Session Types: ☑ All; ☑ hmac-sha1; ☑ hmac-sha256

Default Association Session Type: hmac-sha1

☐ Generate Diffie-Hellman parameters when initiating associations

☑ Enable Provider Authentication Policy Extension (PAPE) 1.0

☐ Send US Government Level of Assurance Policy

☑ Send Private Personal Identifier (PPID) Policy

☐ Send US Government OpenID Trust Level 1 Policy

☐ Send No Personally Identifiable Information (No PII) Policy

Assertion Settings

The assertion setting fields are as follows:

- Map User via Federated Identity - Check this box to specify that the RP will use federated identities to map the incoming assertion to a user record. If not checked, user lookup is based on the assertion data.
- Enable Auto Account Linking - Check this box to specify that the RP/SP should try to map the incoming assertion to a user record when the federated identity does not exist. The mapping uses the assertion data.
- Map User via Attribute Query - Check to enable assertion-to-user mapping through LDAP/RDBMS query, and use the associated Attribute Query field.
- Error when user Mapping fails - Check this box to display a 401 error if mapping of the incoming assertion to a user record using the assertion data fails. If not checked, the RP invokes the authentication engine to authenticate, identify, and provision the user, and the operation resumes after return from the authentication engine. (*Note*: The authentication mechanism configured for the SP that started the flow determines which authentication engine is invoked).
- Require Signed Assertions - Indicates whether the RP/SP requires the incoming assertion to be signed.

Protocol Settings

The protocol setting fields are as follows:

- Enable OpenID 2.0 Protocol - Check this box to enable the OpenID 2.0 protocol.
- Perform OpenID Provider Discovery - Check this box to specify that the RP should discover the OP from discovery URL to initiate the SSO operation. Discovery will determine the features supported by the OP and create the authentication request accordingly. If discovery is disabled, the OpenID provider SSO URL must be

specified and the authentication request format is based solely on the RP configuration.

- Force User Consent - Check this box to prompt the user for consent for any new federation (that is, new ClaimedID) created between the IdP/OP, the RP/SP and the user.
- Default Authentication Mechanism - Holds the local authentication mechanism to use as the authentication method to authenticate the user at the IdP/OP, if the assertion uses the PAPE authentication policy.
- Enabled Session Types - Use the drop-down to list the enabled session types that can be used during the association exchange. Possible values:
  - no-encryption
  - dh-sha1
  - dh-sha256
- Enabled Association Session Types - Use the drop-down to list the enabled association types that the OP supports. Possible values:
  - hmac-sha1
  - hmac-sha256
- Default Association Session Type - Specify the default association session type from one of the enabled types.

Note: Changes in session type and/or association session type require restart of the Oracle Identity Federation server.

- Force User Consent - Check the box to prompt the user for consent for any new federation (that is, new ClaimedID) created between the OP, the RP, and the user.
- Force User Consent Web Context - Contains the Web context of the OpenID consent page to be used instead of the Oracle Identity Federation OpenID built-in consent page.
- Generate Diffie-Hellman parameters when initiating associations - Check the box to generate Diffie-Hellman parameters when initiating association of types `DH-SHA1` or `DH-SHA256`. If not checked, default values are used.
- Enable PAPE 1.0 - Check this box to enable the PAPE 1.0 extension. With this feature you can specify one or more of:
  - US Government Level of Assurance Policy
  - PPID Policy
  - US Government OpenID Trust Level 1 Policy
  - US Government No PII Policy

5.6 Configuring Attribute Sharing with the Oracle Access Manager AuthZ Plug-in

Attribute sharing is a joint feature of Oracle Access Manager and Oracle Identity Federation that implements the SAML Attribute Sharing Profile for X.509 Authentication-Based Systems. In this profile, a user who requests a protected resource or service is authenticated with SSL client X.509 certificates, but authorization is performed with user attributes retrieved from the user's home organization using the SAML protocol. The user's home organization is the identity provider (IdP), and the organization performing authentication and authorization is the service provider (SP).

This section explains how to configure Oracle Access Manager and Oracle Identity Federation for attribute sharing. It contains these topics:

- Components Used for Attribute Sharing
- Remote and Local Users
- Configuring the Oracle Access Manager Plug-ins
- Configuring Oracle Access Manager Schemes and Policies
- Configuring Oracle Identity Federation as an SP Attribute Requester
- Configuring Oracle Identity Federation as an IdP Attribute Responder
- Configuring Oracle Identity Federation for SSL

5.6.1 Components Used for Attribute Sharing

Attribute sharing uses several Oracle Access Manager and Oracle Identity Federation components. The instructions assume that these components have been installed and configured for their normal operation.

Service Provider Components

SP components include:

- Web Server with an Access Manager WebGate - for HTTP requests for a protected URL, performs the SSL client certificate authentication and enforces the access decision from the Oracle Access Manager server
- Oracle Access Manager - performs authentication and authorization for the WebGate. Uses these custom plugins for the attribute sharing feature:
  - authz_attribute Authentication Plug-in - passes the certificate `SubjectDN` to the authz_attribute authorization plug-in
  - authz_attribute Authorization Plug-in - uses the Attribute Requester Service to retrieve attribute values for the user's `SubjectDN` and evaluates a rule expression with the attribute values to determine if access is allowed

  Note: The authentication and authorization plug-ins use the same authz_attribute library.

- Oracle Identity Federation Attribute Requester Service - sends a SAML 1.x or SAML 2.0 attribute query to the IdP Attribute Responder Service determined by the user's SubjectDN, and returns the retrieved attributes to the authz_attribute plug-in.

IdP Component

Oracle Identity Federation Attribute Responder Service or other SAML 1.x or SAML 2.0-compliant federation product - receives a SAML attribute query from the SP Attribute Requester Service, retrieves the attributes for the specified user (subject to local policy controls), and returns a response with the attributes to the Attribute Requester Service.

5.6.2 Remote and Local Users

In addition to remote users authorized by SAML attribute retrieval, the protected resource may also be accessed by local users with attributes defined within the service provider Oracle Access Manager user directory. Local users, configured as discussed here, are detected by the authz_attribute authentication plug-in, which returns a Failure status. The authentication scheme described later uses this status to create a local session for the user, and authorization rules with local LDAP filters can be applied.

5.6.3 Configuring the Oracle Access Manager Plug-ins

Take these steps to configure the Oracle Access Manager plug-ins:

> **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* 10*g* for details about the Web-based user interface.

1. Log in to the Access Server host as the user who installed the Access Server.
2. Create the directory `INSTALLDIR/oblix/config/attributePlug-in`, if it does not already exist.
3. Edit or create the `config.xml` file in the `INSTALLDIR/oblix/config/attributePlug-in` directory, using the sample config.xml file shown here as a template.
4. Edit the attributes and elements of the `config.xml` file as required.
5. Restart the Access Server for changes to take effect.

Sample config.xml

Here is a sample config.xml file:

```
<Config LogLevel="audit" WaitTime="30" SizeLimit="0" MaxConnections="5"
  InitialConnections="2"
    Authn="basic" Username="coreid-as-ashost-6021"  Password="xyzzy"
      KeyPassword="abcde"
    CacheTimeout="3600" MaxCachedUsers="1000" HeaderKeyLength="128"
      RequestFormat="values">
    <Mapping Local="true">
        <DN>O=Company,C=US</DN>
    </Mapping>
    <Mapping URL="https://fed1.company.com:7499/fed/ar/soap">
        <DN>O=PeerA,C=US</DN>
        <DN>O=PeerB,C=US</DN>
    </Mapping>
    <Mapping URL="https://fed2.company.com:7499/fed/ar/soap"
      RequestFormat="all">
        <DN>O=PeerC,C=US</DN>
        <DN>O=PeerD,C=US</DN>
    </Mapping>
    <Mapping URL="https://fed3.company.com:7499/fed/ar/soap">
        <DN>C=US</DN>
    </Mapping>
</Config>
```

Configuration Parameters

The configuration parameters are:

- LogLevel - Controls the amount of information logged to `INSTALL_DIR/oblix/logs/authz_attribute_plug-in_log.txt`.
  - off - Nothing is logged except errors (this is the default).
  - audit - One line is logged for each authentication request, showing the access decision, the user's certificate subject DN or local directory DN, and the HTTP operation and the local part of the requested URL.
  - debug - Logs extensive information useful in debugging problems.
- HTTP connection parameters (authz_attribute plug-in to the Oracle Identity Federation Attribute Requester Service), consisting of:
  - WaitTime - This is the time in seconds to wait for a response; default is 30 seconds.
  - SizeLimit - This is the maximum size in bytes of HTTP messages sent and received (default is unlimited, 0 means unlimited).
  - MaxConnections - This is the maximum number of concurrent HTTP connections (default is 5).
  - InitialConnections - This is the number of current HTTP connections opened initially (default is 2).
- Parameters for authentication of the authz_attribute plug-in to the Oracle Identity Federation Attribute Requester Service, including:
  - Authn - authentication method
    - none - no authentication
    - basic - use HTTP basic authentication with Username and Password (default)
    - cert - use SSL client certificate authentication using key.pem, cert.pem, and KeyPassword
  - Username - This is the username for basic authentication.
  - Password - This is the password for basic authentication.
  - KeyPassword - This is the password for key.pem for SSL client certificate authentication.
- Attribute value cache parameters, including:
  - CacheTimeout - This is the time, in seconds, that cached attribute values will be held before requiring updated values (default 3600 seconds - 1 hour; 0 disables caching).
  - MaxCachedUsers - This is the maximum number of users with cached attribute values; if the cache is full, the least recently used unexpired entries will be reclaimed (default is 1000).
- Mappings of subject DNs to Attribute Requester Service URLs. For each Attribute Requester Service, specify:
  - URL - the URL for the service, of the form `%`*`HTTP_PROTOCOL`*`%://%`*`OIF_HOST`*`%:%`*`OIF_PORT`*`%/fed/ar/soap`, where:
    - `%HTTP_PROTOCOL%` - `http` or `https`

    * *%OIF_HOST%:%OIF_PORT%* - This is the host and port of Oracle Identity Federation.

    For example: `https://fed1.company.com:7499/fed/ar/soap`

  – Local - if `true`, the matching users are local and an Attribute Requester Service is not used. If `true`, the URL parameter is ignored

  – DN - one or more elements specifying a DN pattern to match against the user Subject DN; the pattern is simply the right most components of the DN. For example: `O=PeerA,C=US`

- Attribute query properties - The `RequestFormat` parameter determines the attributes and values returned in an attribute response. RequestFormat overrides authorization rules; for example, if an authorization rule specifies both attributes and values, but `RequestFormat` specifies names, the query omits values. RequestFormat can be specified with these options:

  – RequestFormat="values"

    The `AttributeQuery` contains attribute names and values taken from the authorization rule's `ruleExpression`. The Attribute Responder will only return user attributes and values that are in the `AttributeQuery`. This is the default setting. This setting minimizes the amount of memory used for cached attribute values (values are only requested when needed for authorization), at the cost of more frequent attribute requests.

  – RequestFormat="names"

    The `AttributeQuery` contains attribute names but not values taken from the ruleExpression. The Attribute Responder returns all the user's values for the named attributes, subject to any Responder policies controlling access to the attributes values. This setting provides a trade-off between cache memory usage and attribute requests that is somewhere between the "values" and "all" setttings. Note: With this setting, the `AttributeQuery` does not disclose to the IdP what attribute values are required for authorization; for security reasons, this might be preferred over the "values" setting.

  – RequestFormat="all"

    The `AttributeQuery` does not contain any attribute names or values. The Attribute Responder returns all the attributes and values for the user subject to any Responder policies controlling access to the attributes values. This setting minimizes the number of attribute requests (only one request per user), at the cost of more memory used for caching attribute values before they are used (and may never be used) for authorization. This setting works best when the Attribute Responder policies have been reasonably configured to return only attributes that the SP might want. Note: With this setting, the `AttributeQuery` does not disclose to the IdP what attributes are required for authorization; for security reasons, you may prefer this over the "values" and "names" settings.

As illustrated in the sample config.xml file, the RequestFormat parameter can appear in the `<Config>` element, where it sets the default request format, and in the `<Mapping>` elements, where it sets the request format for subject DNs covered by the mappings.

Mapping Examples for the Sample Configuration

Here are some mapping examples for the sample `config.xml` configuration file shown earlier.

Table 5–7 Mapping Examples for config.xml

User Subject DN	Maps to URL
E=john.smith@company.com,CN=John Smith, OU=Development,O=Company,C=US	local
E=betty.jones@peera.com.CN=Betty Jones,OU=Marketing,O=PeerA,C=US	https://fed1.company.com:7499/fed/ar/soap
E=sally.smith@peerd.com,CN=Sally Smith,OU=Marketing,O=PeerD,C=US	https://fed2.company.com:7499/fed/ar/soap
E=bill.jones@peerx.com,CN=Bill Jones,OU=Finance,O=PeerX,C=US	https://fed3.company.com:7499/fed/ar/soap

Configuring SSL and Client Certificate Authentication

Use these steps to configure HTTPS and SSL client certificate authentication:

1. If HTTPS is used between the authz_attribute plug-in and at least one Attribute Requester Service, set up the trusted CA list in `INSTALL_DIR/oblix/config/attributePlug-in/cacerts.pem`. For each CA that certifies an Attribute Requester service, add the PEM formatted certificate (including the `-----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----`) to `cacerts.pem`.
2. If SSL client certificate authentication is used between the authz_attribute plug-in and at least one Attribute Requester Service, set up the `key.pem` and `cert.pem` files:
   - Generate the private key and certificate request using the openssl utility included with Oracle Access Manager with these steps:
     – `cd INSTALL_DIR/oblix/tools/openssl`
     – `openssl req -config openssl.cnf -newkey rsa:1024 -keyout../../config/attributePlug-in/key.pem -out../../config/attributePlug-in/req.pem`
   - Send `INSTALL_DIR/oblix/config/attributePlug-in/req.pem` to your CA to get a certificate.
   - Copy the generated certificate to `INSTALL_DIR/oblix/config/attributePlug-in/cert.pem`.
3. Restart the Access Server to ensure that the plug-in uses the `PEM` files.

5.6.4 Configuring Oracle Access Manager Schemes and Policies

This section explains how to configure Oracle Access Manager schemes and policies for Oracle Identity Federation. It contains these sections:

- Configuring the Attribute Sharing Authentication Scheme
- Configuring the Attribute Sharing Authorization Scheme
- Configuring an Oracle Access Manager Policy using Attribute Sharing

5.6.4.1 Configuring the Attribute Sharing Authentication Scheme

Take these steps:

> **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* 10*g* for details about the Web-based user interface.

1. Log in to the Oracle Access Manager System Console as a Master Access Administrator. Select the Access System Configuration panel and Authentication Management.
2. Click **Add** and fill out the Define a New Authentication Scheme form.
   - Name: OIF Attribute Sharing
   - Description: Performs an SSL client certificate authentication for Oracle Identity Federation Attribute Sharing authorization
   - Level: set based on the requirements of the protected resources; should be higher than any password schemes
   - Challenge Method: X509Cert
   - Challenge Parameter: ssoCookie:Expires=Tue, 1 Nov 2005 00:00:00 GMT

   > **Note:** To ensure that this authentication scheme is run on every access to protected resources, this challenge parameter forces the browser to discard the `ObSSOCookie`, which forces Oracle Access Manager to re-authenticate.

   - SSL Required: yes
   - Enabled: no (until the plug-ins are configured)

   Click **Save** and commit the changes.
3. Select the Plug-ins tab and click **Modify**. Add the plug-ins and parameters shown in the table. To enter built-in plug-ins, select the plug-in name from the drop-down list.To enter custom plug-ins, select Custom Plug-in from the drop-down list and enter the plug-in name in the text box. Click **Save** when all plug-ins have been added.

Plug-in Name	Type	Plug-in Parameters
authz_attribute	custom	(none)
cert_decode	built-in	(none)
credential_mapping	built-in	obMappingBase="MAPPING_ BASE",obMappingFilter="(uid=OblixAnon ymous)"
credential_mapping	built-in	obMappingBase="MAPPING_ BASE",obMappingFilter="(&(&(objectclass =PERSON_OBJECTCLASS) (USER_ ATTRIBUTE=%certSubject.FIELD%))(|(!(o buseraccountcontrol=*))(obuseraccountcont rol=ACTIVATED)))"

Here is an example:

Plug-in Name	Plug-in Parameters
authz_attribute	
cert_decode	
credential_mapping	obMappingBase="o=Company,c=US", obMappingFilter="(uid=OblixAnonymous)"
credential_mapping	obMappingBase="o=Company,c=US", obMappingFilter="(&(&(objectclass=inetorgperson)(mail=%certSubject.E%)) (|(!(obuseraccountcontrol=*))(obuseraccountcontrol =ACTIVATED)))"

4. Select the Steps panel. Add the following steps:

Step Name	Active Plug-ins	Purpose
SubjectDN	authz_attribute	Extracts SubjectDN from certificate; determines if user is remote or local
RemoteUser	first credential_mapping	Creates an anonymous session for a remote user
LocalUser	cert_decode, second credential_mapping	Creates a real session for a local user

5. Select the **Authentication Flow** panel, click **Modify** and set the flow shown in the table. Note: The authz_attribute plug-in returns `Success` if the user is remote and `Failure` if the user is local.

Step name	Initiating Step	On Success Next Step	On Failure Next Step
Default Step	no	Stop	Stop
SubjectDN	yes	RemoteUser	LocalUser
RemoteUser	no	Stop	Stop
LocalUser	no	Stop	Stop

6. Return to the **Steps** panel and remove the now-unused Default step.
7. Return to the **General** panel and enable the authentication scheme.

5.6.4.2 Configuring the Attribute Sharing Authorization Scheme

Take these steps to configure the attribute sharing authorization scheme:

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* for details about the Web-based user interface.

1. Log in to the Oracle Access Manager System Console as a Master Access Administrator. Select the Access System Configuration panel and Authorization Management.
2. Click **Add** and fill out the Define a new Authorization Scheme form:
   - Name: OIF Attribute Sharing
   - Description: Uses Oracle Identity Federation to obtain attributes for remote users to evaluate the rule expression

- Shared Library: oblix/lib/authz_attribute
- Plug-in is Managed Code: no
- Managed Code Name Space: (none)
- User Parameter: RA_SubjectDN (Note: This uses the "reverse action" feature to obtain the SubjectDN header set by the authz_attribute plug-in.)
- Required Parameter
  - Name: ruleExpression
  - Value: (none) (Note: Each access policy authorization rule will supply the rule expression.)
- Click Save to commit the changes.

5.6.4.3 Configuring an Oracle Access Manager Policy using Attribute Sharing

Take these steps to configure an Oracle Access Manager policy using the Attribute Sharing profile:

1. Log in to Oracle Access Manager as a Master or Delegated Access Administrator. Select Create Policy Domain.
2. Fill out the General panel form:
   - Name: as appropriate (for example, Oracle Identity Federation Attribute Sharing Test)
   - Description: as appropriate

   Click **Save**.
3. Select the Resource panel and add one or more resource URL prefixes to protect (for example, `/attribute-test`).
4. Select the Authorization Rules panel and add an authorization rule for each set of attributes (represented as a rule expression) required for a remote user.
   - Select Custom Authorization Scheme and click **Add**.
   - Fill out the authorization rule form and click **Save**.
     - Name: as appropriate (for example, Peer Marketing VP)
     - Description: as appropriate
     - Authorization Scheme: OIF Attribute Sharing
   - Select the Plug-in Parameters panel, click **Modify**, and set the ruleExpression parameter as specified in the table. Note: White space is allowed around =, !=, &, and `|`.

Element	Syntax	Meaning
none	alphanumeric string including '-', '_', and '.'	Name of attribute to request from the user's identity provider

Element	Syntax	Meaning
value	one of "string", any, or null	Required attribute value. With Oracle COREid Access 7.0.4 the string is restricted to Latin-1 characters. With Oracle Access Manager 10.1.4 and later, the string can contain any Unicode characters. The any value retrieves and matches all values for the attribute. The null value matches a SAML <Attribute> with the xsi:nil="true" attribute.
comparison	name = value, name != value, or (expression)	True if the user has/does not have the attribute value
and-clause	comparison & comparison	True if both comparisons are true.
or-clause	comparison | comparison	True if either comparison is true. & has higher precedence than !.

- Name examples:
  - title = "VP" & function = "Marketing"
  - title = "VP" | title = "Director"
  - title = "VP" & (function = "Marketing" | function = "Finance")
  - title = any & function = any
- Set any timing conditions or actions as desired for the authorization rule.
- Return to the **General** panel and enable the rule.

5. Select the Authorization Rules panel and add an authorization rule for any local user attributes.
   - Select Oracle Authorization Scheme and click **Add**.
   - Fill out the authorization rule form:
     - Name: as appropriate (for example, Company Marketing VP).
     - Description: as appropriate
     - Enabled: yes
     - Allow Takes Precedence: no

     Click **Save**.
   - Select the **Allow Access** panel, click **Modify**, and add an LDAP filter for the local attributes. You can use the Query Builder in the Oracle Access Manager Identity User Manager (**Configuration**, then **Delegate Administration**, then **Build Filter**). For example:

     ```
     ldap:///o=Company,c=US??sub?(&(title=VP)(function=Marketing))
     ```
   - Set any timing conditions or actions as desired for the authorization rule.
   - Return to the **General** panel and enable the rule.
6. Select the **Default Rules** panel and add the default authentication rule:
   - Name: as appropriate

- Description: as appropriate
- Authentication Scheme: OIF Attribute Sharing

Click **Save**.

7. Select the **Authorization Expression** panel and add the default authorization rule:
   - Select the applicable remote authorization rule as defined above and click **Add** (for example, Peer Marketing VP).
   - If there is a corresponding local authorization rule, select OR and add the local authorization rule. (for example, Peer Marketing VP | Company Marketing VP).

   Click **Save**.
8. Alternatively, you can add policies to the policy domain with authorization expressions for subsets of the protected URLs.

5.6.5 Configuring Oracle Identity Federation as an SP Attribute Requester

Take these steps to configure Oracle Identity Federation as an attribute requester in service provider mode:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Enable the Attribute Requester functionality:
   - Navigate to **Administration**, then **Service Provider**.
   - Check the **Enable Attribute Requester Service** box, and click **Apply**.

   Note: Checking the Enable Attribute Requester Service box enables the Attribute Requester feature. It also modifies the SP's metadata to include information about the Attribute Requester service. Note that the metadata at the peer providers' sites must be updated with the new version.
3. Upload the SAML 1.x or SAML 2.0 IdP metadata, or manually create an entry for a SAML 1.x provider.
   - Navigate to **Administration**, then **Federations**.
   - Click **Add**.
   - To upload SAML 1.x or SAML 2.0 metadata, select **Upload Metadata** and enter the location of the IdP metadata and an additional description.
   - To add a SAML 1.x provider manually, select **Add Trusted Provider Manually**, and enter the Provider ID, the Provider Version (SAML 1.1 or SAML 1.0), select Identity Provider and Attribute Responder as the Provider Type, and enter an additional description.
   - Click **OK**.
4. Configure the DN to IdP mapping:
   - Navigate to **Administration**, then **Service Provider**.
   - Click **Configure Attribute Requester Service**.

- Select the Default Attribute Authority from the drop down list, and click **Apply**.
- To add a mapping:
  - Click **Add**.
  - Enter the DN or sub-DN (for example, c=us)
  - Map this DN or sub-DN to an existing IdP
  - Repeat the operation if necessary
- Click **OK**.

5. Enable and configure certificate Validation:
   - Navigate to **Administration**, then **Security and Trust.**
   - Select **Enable Certificate Validation**, and click **Apply**.
   - Add Trusted CAs or CRLs by clicking Add in the corresponding table and selecting the location of the CA or the CRL. (Note: if certificate Validation is enabled, a Trusted CA is required to validate signatures).

   Note: Configuring DN to IdP and certificate Validation is optional.

6. If using SAML 2.0, enable encryption:
   - Navigate to **Administration**, then **Service Provider**.
   - In the **SAML 2.0** tab, under **Protocol Settings**:
     - Check **Send Encrypted NameIDs** to encrypt the Name Identifiers in the `AttributeQuery` to the Attribute Responder.
     - Check **Send Encrypted Attributes** to encrypt the Attributes in the AttributeQuery to the Attribute Responder.
   - Click **Apply**.

   Note: Encryption is optional.

7. The Attribute Requester service is available at `http://sp-hostname:port/fed/ar/soap`.

After enabling the attribute requester capabilities and setting the Default Attribute Authority and/or the DN Mappings, you must configure the attribute name mappings and the attribute value mappings. See Section 5.9.2, "Mapping and Filtering Configuration" for more information.

Additional topics include:

- If Using HTTP Basic Authentication With OHS
- If Using HTTP Basic Authentication Without OHS
- If Using SSL Client Authentication

5.6.5.1 If Using HTTP Basic Authentication With OHS

If using basic authentication between the plug-in and Oracle Identity Federation, you need to add the following to the httpd.conf file of the OHS for your Oracle Identity Federation instance:

```
<LocationMatch "/fed/ar/soap">
    AllowOverride None
    AuthType Basic
    AuthName "Restricted Files"
    AuthUserFile /private/oifpassword
    Require user alice
    Order allow,deny
    Allow from all
</LocationMatch>
```

A user passwords file must also be created using the `htpasswd` utility. In the above example, the AuthUserFile containing the users and their passwords points to the `/private/oifpassword` file, in which the user alice is defined.

This example creates such a file by adding the user `alice`:

```
$ORACLE_HOME/ohs/bin/htpasswd -c /private/oifpassword alice
```

5.6.5.2 If Using HTTP Basic Authentication Without OHS

If using HTTP Basic Authentication without Oracle HTTP Server, see Section 6.9.2, "HTTP Basic Authentication".

5.6.5.3 If Using SSL Client Authentication

If using client certificate authentication, see Section 8.1, "Configuring SSL for Oracle Identity Federation".

5.6.6 Configuring Oracle Identity Federation as an IdP Attribute Responder

Take these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Enable the Attribute Responder functionality:
   - Navigate to **Administration**, then **Identity Provider**.
   - In the SAML 2.0 or SAML 1.x tab, select **Enable Attribute Query Responder**. If using SAML 2.0, select **Use Identity Federation for Attribute Response** if you want the user in the attribute request to be located in the IdP using its federated identity. Note that if using this setting, the user must have a federation identity and its Name ID value and format must match the subject value and format specified in the `AttributeQuery`.
   - Click **Apply**.

   Note: Checking the **Attribute Responder Enabled** box enables the attribute authority feature. It also modifies the IdP's metadata to include information about the attribute authority service. Note that the metadata at the peer providers' sites must be updated with the new version.

3. Map the Name ID Formats:
   - Navigate to **Administration**, then **Identity Provider**.
   - In the SAML 2.0 tab, under Assertion settings, check that the subject formats to be used are enabled and mapped to the correct user entry attribute from the user repository:
     – Use `dn` to map the X.509 Subject Name to an entry's Distinguished Name, or use any attribute from a user entry.

   Note: The attribute selected for the X509 Subject Name must exactly match the client certificate subject DN, following the format specified in RFC 2253. If unsure of the format, you can perform a test with the SP and look at the Subject NameIdentifier value sent from the SP, which is logged in.

   - Click **Apply**.
4. Upload the SAML 1.x or SAML 2.0 IdP metadata, or manually create an entry for a SAML 1.x provider
   - Navigate to **Administration**, then **Federations**.
   - Click **Add**
   - To upload SAML 1.x or SAML 2.0 metadata, select **Upload Metadata** and enter the location of the IdP metadata and an additional description.
   - To add a SAML 1.x provider manually, select **Add Trusted Provider Manually**, and enter the Provider ID, the Provider Version (SAML 1.1 or SAML 1.0), select Service Provider and Attribute Requester as the Provider Type, and enter an additional description.
   - Click **Apply**.
5. Configure the Attribute Mappings for the SP Attribute Requester:
   - Navigate to **Administration**, then **Federations**.
   - Select the SP Attribute Requester entry and click **Edit**.
   - Select **Update Manually**, and under Oracle Identity Federation Settings, click **Edit Attribute Mappings and Filters**.
   - To add an attribute mapping:
     – Click **Add**.
     – Enter the user repository attribute name in the User Attr Name column.
     – In Assertion Attr Name, enter the identifier used in the AttributeQuery or assertion to reference the attribute.
     – Enter the Format or Namespace, if any. This is an optional field used to specify the format or the namespace of the SAML attribute, depending on the version.

       - For SAML 1.x, this field's value is used to set the SAML attribute's namespace.

       - For SAML 2.0, this value is used to set the SAML attribute's NameFormat; if this field is empty, the NameFormat of the SAML attribute will be set to urn:oasis:names:tc:SAML:2.0:attrname-format:basic; otherwise the NameFormat will hold the value specified in this field.

   – Repeat the operation to add other attribute mappings.

- Click **OK**.

Note: For an SP using Oracle Identity Federation, the assertion Attr Name is determined by the attribute name in a ruleExpression as set in Section 5.6.4.3, "Configuring an Oracle Access Manager Policy using Attribute Sharing". The attribute names must be agreed upon between the IdP and SP.

---

6. Enable and configure certificate validation:
   - Navigate to **Administration**, then **Security and Trust**
   - Select **Enable Certificate Validation**, and click **Apply**.
   - Add Trusted CAs or CRLs by clicking Add in the corresponding table and selecting the location of the CA or the CRL. (Note: if certificate Validation is enabled, a Trusted CA is required to validate signatures).

Note: Configuring certificate Validation is optional.

---

7. If using SAML 2.0, enable encryption:
   - Navigate to **Administration**, then **Service Provider**.
   - In the **SAML 2.0** tab, under Protocol Settings:
     – Check **Send Encrypted NameIDs** to encrypt the Name Identifiers in the `AttributeQuery` to the Attribute Responder.
     – Check **Send Encrypted Attributes** to encrypt the Attributes in the `AttributeQuery` to the Attribute Responder.
   - Click Apply.

Note: Encryption is optional.

After enabling the attribute responder capability, you must configure:

- which attributes to send
- attribute name mappings
- attribute value mappings
- attribute value filters

See Section 5.9, "Configuring Attribute Mapping and Filtering" for more information.

5.6.7 Configuring Oracle Identity Federation for SSL

To configure SSL for the server, see Section 8.1, "Configuring SSL for Oracle Identity Federation".

5.7 Configuring Identity Provider to send attributes in SSO Assertions

During a Single Sign-On operation, the identity provider can optionally include attributes in the authentication assertion to be consumed by the service provider.

Take these steps to enable attributes to be sent in an assertion:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Federations**.
3. Select the service provider with which you want to configure attribute sharing, and click **Edit**.
4. Select **Update Manually**.
5. Under the Oracle Identity Federation Settings tab, check **Enable Attributes in Single Sign-On**.
6. Below, check the boxes to specify the Name ID formats for which attributes will be sent in assertions.
7. Click **Apply**.

After checking the **Enable Attributes in Single Sign-On** box, you need to configure:

- the attributes to send
- attribute name mapping
- attribute value mappings
- attribute value filters

See Section 5.9, "Configuring Attribute Mapping and Filtering" for more information.

5.8 Web Services Interface for Attribute Sharing

This section describes the Oracle Identity Federation's Attribute Requester Service Interface. It contains these topics:

- Overview of the Service Interface
- Attribute Request Message
- Attribute Response Message
- Interface WSDL

5.8.1 Overview of the Service Interface

The Attribute Requester Service provides a request/response interface using the SOAP POST protocol. The service supports the X.509 authn-based attribute sharing profile and follows the SAML `<AttributeQuery>` convention.

The service can be invoked to send `samlp:AttributeQuery` messages to a remote identity provider.

Here are the steps exercised when the web service client sends an `AttributeRequest` to the Oracle Identity Federation/Attribute Requester server:

1. The web service client sends an AttributeRequest message using the SOAP protocol.

2. Oracle Identity Federation processes the incoming `AttributeRequest` message, and selects the IdP to which to send the SAML `AttributeQuery`, based either on the IdP specified on the Request, or on the `Subject` contained in the `AttributeRequest`.
3. Oracle Identity Federation applies, for the specific remote IdP, the attribute value mapping for the optional attribute values listed in the AttributeRequest.
4. Oracle Identity Federation applies, for the specific remote IdP, the attribute name mapping for the optional attribute listed in the AttributeRequest.
5. Oracle Identity Federation sends the `AttributeQuery` to the remote IdP.
6. Oracle Identity Federation receives the response containing the assertion, along with the attributes sent by the IdP.
7. Oracle Identity Federation applies, for the specific remote IdP, the attribute name mapping for the attribute names listed in the assertion's `AttributeStatement`.
8. Oracle Identity Federation applies, for the specific remote IdP, the attribute value mapping for the attribute values listed in the assertion's `AttributeStatement`.
9. Oracle Identity Federation builds the `AttributeResponse` message, and returns it to the web service client in a SOAP response message.

5.8.2 Attribute Request Message

The AttributeRequest message issues a request for attribute data about a user. The AttributeRequest specifies these inputs:

- The Subject: A string representing the user. This is a required input.
- The Subject Format: A URI specifying how the Subject string represents the user. If not present, format "`oracle:security:nameid:format:x509`" will be used. Valid formats are:
  - `oracle:security:nameid:format:x509`: Indicates that the Name ID is the Subject DN.
  - `oracle:security:nameid:format:entity`: Indicates that the Name ID is the identifier of an entity that provides SAML services.   This Name ID Format only applies to the SAML 2.0 protocol.
  - `oracle:security:nameid:format:emailaddress:` Indicates that the Name ID is in the form of an email address.
  - `oracle:security:nameid:format:windowsdomainqualifiedname`: Indicates that the Name ID is a Windows domain qualified name (A Windows domain qualified name is a string of the form "DomainName\UserName", where the DomainName and "\" can be omitted).
  - `oracle:security:nameid:format:kerberos`: Indicates that the Name ID is in the form of a Kerberos principal name using the format name`[/instance]@REALM`. This Name ID Format only applies to the SAML 2.0 protocol.
  - `oracle:security:nameid:format:persistent`: Indicates that the Name ID is a persistent opaque identifier for the user that is specific to an IdP and SP. This Name ID Format only applies to the SAML 2.0 protocol.
  - `oracle:security:nameid:format:transient`: Indicates that the Name ID is an opaque and temporary identifier for the user. This Name ID Format only applies to the SAML 2.0 protocol.

- `oracle:security:nameid:format:unspecified`: Indicates that the interpretation of the Name ID is left up to the implementation.
- `oracle:security:nameid:format:custom`: Indicates that the Name ID is a custom value.
- `oracle:security:nameid:format:userid`: Indicates that the Name ID is the User ID used by Oracle Identity Federation to identify the user.

Note: To enable/disable Name ID formats and map them to attributes in the user data store, follow these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance
2. Navigate to **Administration**, then **Identity Provider** or **Service Provider** (to configure IdP and SP, respectively).
3. In the SAML 2.0/SAML 1.X tabs, modify the Assertion Subject NameID Formats by:
   a. Clicking the **Enabled** box next to the formats you wish to enable.
   b. Mapping each format to an attribute in the user data store.
4. Click **Apply**.

- The attribute authority to which the `AttributeQuery` is to be sent. If no attribute authority is specified, Oracle Identity Federation will determine what attribute authority to send the `AttributeQuery` as follows:
  - If the Subject Format is "`oracle:security:nameid:format:x509`", or if it is not present, Oracle Identity Federation will map the `Subject` value to an identity provider. If no mapping is found for the `SubjectDN`, the default attribute authority is used.
  - Otherwise, Oracle Identity Federation will use the default attribute authority.

  See Also: Section 5.6.5, "Configuring Oracle Identity Federation as an SP Attribute Requester" for instructions on how to configure the default attribute authority and the SubjectDN to IdP mappings

- Zero or more attributes to be retrieved for the user.
- For each attribute, zero or more values. A NULL value can be represented as `<Value Null="true"/>`.

The AttributeRequest message is wrapped in a SOAP Envelope and Body and sent in an HTTP POST request. Examples of AttributeRequest messages follow.

Example 1

In the following request, the `Subject` format is not specified and is therefore assumed to be "`oracle:security:nameid:format:x509`". The target IdP is also not specified and so Oracle Identity Federation will determine the attribute authority to use by mapping the `SubjectDN` to an IdP.

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <orafed-arxs:AttributeRequest
```

```
         xmlns:orafed-arxs=http:"//www.oracle.com/fed/ar/10gR3">
          <orafed-arxs:Subject>cn=alice,cn=users,dc=us,dc=oracle,dc=com
          </orafed-arxs:Subject>
          <orafed-arxs:Attribute Name="mail">
             <orafed-arxs:Value>alice@oracle.com</orafed-arxs:Value>
             <orafed-arxs:Value>bob@oracle.com</orafed-arxs:Value>
          </orafed-arxs:Attribute>
          <orafed-arxs:Attribute Name="firstname">
             <orafed-arxs:Value>Bobby</orafed-arxs:Value>
             <orafed-arxs:Value>Charles</orafed-arxs:Value>
          </orafed-arxs:Attribute>
          <orafed-arxs:Attribute Name="lastname">
          </orafed-arxs:Attribute>
       </orafed-arxs:AttributeRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 2

In the following request, the target IdP is specified to be "`http://my-corp.com/fed/idp`", so Oracle Identity Federation will send the AttributeQuery to this attribute authority. Also, the Subject Format is "`oracle:security:nameid:format:userid`", so the Subject value "alice" is taken to be the User ID of the user of which attributes are requested.

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <orafed-arxs:AttributeRequest
xmlns:orafed-arxs=http://www.oracle.com/fed/ar/10gR3
TargetIDP="http://my-corp.com/fed/idp">
         <orafed-arxs:Subject Format="oracle:security:nameid:format:userid">alice
         </orafed-arxs:Subject>
         <orafed-arxs:Attribute Name="mail">
            <orafed-arxs:Value>alice@oracle.com</orafed-arxs:Value>
            <orafed-arxs:Value>bob@oracle.com</orafed-arxs:Value>
         </orafed-arxs:Attribute>
         <orafed-arxs:Attribute Name="firstname">
            <orafed-arxs:Value>Bobby</orafed-arxs:Value>
            <orafed-arxs:Value>Charles</orafed-arxs:Value>
         </orafed-arxs:Attribute>
         <orafed-arxs:Attribute Name="lastname">
         </orafed-arxs:Attribute>
      </orafed-arxs:AttributeRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The output rules are as follows:

- Following the SAML `<AttributeQuery>` convention, if no attributes are named, all of the user's attributes are returned.
- If one or more attributes are named in the request, only these are returned.
- If values are specified in the request, the attribute authority will only return a local attribute value if the value is present in the request.
- Attributes are returned subject to the responder's local policy.

5.8.3 Attribute Response Message

The Attribute Requester service returns the AttributeResponse message to a SOAP client following an attribute request.

Outputs of `AttributeResponse` include:

- the status of the SAML 1.x or SAML 2.0 query (`Success` or `Failure`, with the reason). The client can use this information for logging.
- the `Subject`, as specified in the Request
- the Subject Format, as specified in the Request
- zero or more `<Attribute>` elements, with each element supplying an attribute name and zero or more values

Note the following about returned attribute values:

- All values are UTF-8 strings.
- Following the SAML `AttributeQuery` convention, if the requestor is not allowed to see any values for an attribute, the Attribute element will be returned with no Value elements.
- An attribute value of `NULL` is represented by `<Value Null="true"/>`.
- The `CacheFor` attribute in the `AttributeResponse` message specifies how long the attribute values can be cached.

The `AttributeResponse` message is wrapped in a SOAP Envelope and Body and returned in an HTTP 200 OK response. The following attribute responses could correspond to the attribute requests in the examples above:

Example 1

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
      <orafed-arxs:AttributeResponse
      xmlns:orafed-arxs="http://www.oracle.com/fed/ar/10gR3" CacheFor="1199">
         <orafed-arxs:Status>Success</orafed-arxs:Status>
         <orafed-arxs:Subject>cn=alice,cn=users,dc=us,dc=oracle,dc=com
         </orafed-arxs:Subject>
         <orafed-arxs:Attribute Name="lastname">
            <orafed-arxs:Value>Appleton</orafed-arxs:Value>
         </orafed-arxs:Attribute>
         <orafed-arxs:Attribute Name="firstname"></orafed-arxs:Attribute>
         <orafed-arxs:Attribute Name="mail">
            <orafed-arxs:Value>alice@oracle.com</orafed-arxs:Value>
         </orafed-arxs:Attribute>
      </orafed-arxs:AttributeResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 2

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
      <orafed-arxs:AttributeResponse
xmlns:orafed-arxs="http://www.oracle.com/fed/ar/10gR3" CacheFor="1199">
```

```
         <orafed-arxs:Status>Success</orafed-arxs:Status>
         <orafed-arxs:Subject Format="oracle:security:nameid:format:userid">alice
         </orafed-arxs:Subject>
         <orafed-arxs:Attribute Name="lastname">
            <orafed-arxs:Value>Appleton</orafed-arxs:Value>
         </orafed-arxs:Attribute>
         <orafed-arxs:Attribute Name="firstname"></orafed-arxs:Attribute>
         <orafed-arxs:Attribute Name="mail">
            <orafed-arxs:Value>alice@oracle.com</orafed-arxs:Value>
         </orafed-arxs:Attribute>
      </orafed-arxs:AttributeResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5.8.4 Interface WSDL

The WSDL that formally defines the attribute requester service interface is as follows:

```
<?xml version ="1.0" encoding="US-ASCII" ?>
<wsdl:definitions name="AttributeRequesterFed"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                    xmlns:xml="http://www.w3.org/XML/1998/namespace"
                    xmlns:orafed-arxs="http://www.oracle.com/fed/ar/10gR3"
                    xmlns:orafed-arwsdl="http://www.oracle.com/fed/ar/wsdl"
                    targetNamespace="http://www.oracle.com/fed/ar/wsdl">
        <wsdl:types>
                <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                        targetNamespace="http://www.oracle.com/fed/ar/10gR3"
                        elementFormDefault="qualified"
                        attributeFormDefault="unqualified">
                        <xs:complexType name="SubjectType">
                            <xs:simpleContent>
                                <xs:extension base="xs:string">
                                    <xs:attribute name="Format" type="xs:string"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                        <xs:element name="Subject"
type="orafed-arxs:SubjectType"/>

                        <xs:complexType name="ValueType">
                            <xs:simpleContent>
                                <xs:extension base="xs:string">
                                    <xs:attribute name="Null" type="xs:boolean"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                        <xs:element name="Value" type="orafed-arxs:ValueType"/>

                        <xs:complexType name="AttributeType">
                           <xs:attribute name="Name" type="xs:ID"/>
                           <xs:sequence>
                              <xs:element ref="orafed-arxs:Value"
minOccurs="0" maxOccurs="unbounded"/>
                           </xs:sequence>
                        </xs:complexType>
                        <xs:element name="Attribute"
type="orafed-arxs:AttributeType"/>
```

```
                        <xs:complexType name="AttributeRequestType">
<xs:attribute name="TargetIDP" type="xs:string"/>
                                <xs:sequence>
                                        <xs:element ref="orafed-arxs:Subject"/>
                                        <xs:element ref="orafed-arxs:Attribute"
minOccurs="0" maxOccurs="unbounded"/>
                                </xs:sequence>
                        </xs:complexType>
                        <xs:element name="AttributeRequest"
type="orafed-arxs:AttributeRequestType"/>

                        <xs:complexType name="AttributeResponseType">
                                <xs:sequence>
                                        <xs:element name="Status"
type="xs:string"/>
                                        <xs:element ref="orafed-arxs:Subject"/>
                                        <xs:element ref="orafed-arxs:Attribute"
minOccurs="0" maxOccurs="unbounded"/>
                                </xs:sequence>
                                <xs:attribute name="CacheFor"
type="xs:unsignedInt"/>
                        </xs:complexType>
                        <xs:element name="AttributeResponse"
type="orafed-arxs:AttributeResponseType"/>
                </xs:schema>
        </wsdl:types>

        <wsdl:message name="AttributeRequestMessage">
                <wsdl:part name="body" element="orafed-arxs:AttributeRequest"/>
        </wsdl:message>

        <wsdl:message name="AttributeResponseMessage">
                <wsdl:part name="body" element="orafed-arxs:AttributeResponse"/>
        </wsdl:message>

        <wsdl:portType name="AttributeRequesterServicePortType">
                <wsdl:operation name="AttributeRequestOp">
                        <wsdl:input
message="orafed-arwsdl:AttributeRequestMessage"/>
                        <wsdl:output
message="orafed-arwsdl:AttributeResponseMessage"/>
                </wsdl:operation>
        </wsdl:portType>

        <wsdl:binding name="AttributeRequesterServiceBinding"
type="orafed-arwsdl:AttributeRequesterServicePortType">
                <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
                <wsdl:operation name="AttributeRequestOp">
                        <soap:operation
soapAction="http://www.oracle.com/fed/AttributeRequestOp" />
                        <wsdl:input>
                                <soap:body use="literal"/>
                        </wsdl:input>
                        <wsdl:output>
                                <soap:body use="literal"/>
                        </wsdl:output>
                </wsdl:operation>
        </wsdl:binding>
        <wsdl:service name="AttributeRequesterService">
                <wsdl:port name="AttributeRequesterServicePort"

binding="orafed-arwsdl:AttributeRequesterServiceBinding">
                        <soap:address
location="http://node.us.example.com:1234/fed/ar/soap"/>
                </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

The types and message sections define the contents of the AttributeRequest and AttributeResponse messages.

The built-in XML Scheme type ID is used for the Name attribute of the attribute elements; this type approximates the desired syntax for attribute names (letters, numbers, "'_", "-", and ".") However, ID (which is derived from the XML NCName type) also includes a number of Unicode combining characters and extenders.

See Also: The W3C specification, Namespaces in XML, at http://www.w3.org/TR/1999/REC-xml-names-19990114/#NT-NCName

The binding and service sections specify how the messages are to be sent over SOAP and HTTP(S).

5.9 Configuring Attribute Mapping and Filtering

This section explains how to configure the attribute mapping functionality in Oracle Identity Federation. It contains these topics:

- Introduction to Attribute Mapping and Filtering
- Mapping and Filtering Configuration

5.9.1 Introduction to Attribute Mapping and Filtering

Supported Entities

Oracle Identity Federation supports attribute mapping for the following:

- Attribute Authority
- Attribute Requester
- Identity Provider, when sending attributes in SSO assertions

Mapping Capabilities

Oracle Identity Federation provides the following attribute mapping capabilities:

- Attribute Name Mapping: maps local attribute names to external attribute names used in SAML messages
- Attribute Value Mapping: maps local attribute values to external attribute values used in SAML messages
- Attribute Value Filtering: filters local attribute values by sending only allowed values in assertion messages

---

Note: In 11*g* Release 1 (11.1.1), all attribute mapping and filtering is available only as per-peer-provider configuration, not at the global level.

Attribute Sources

Oracle Identity Federation can map attributes to an outgoing assertion from these sources:

1. user sessions
2. user data stores
3. static values from the Oracle Identity Federation configuration

This section contains these topics:

- Attribute Name Mapping
- Attribute Value Mapping
- Attribute Value Filtering

5.9.1.1 Attribute Name Mapping

Attribute name mapping allows the administrator to specify the name with which a local attribute should be defined in the SAML messages when sending or receiving messages.

On the IdP/Attribute Authority side, when a mapping is defined, Oracle Identity Federation can also be configured to send the attribute to a specific peer provider. Thus, when no name mappings are defined, Oracle Identity Federation is configured to send no attributes to peer providers.

Oracle Identity Federation exercises attribute name mapping when acting as a:

- Attribute Authority
- Attribute Requester
- Identity Provider, when sending attributes in SSO assertions

One-to-One Mapping

Attribute name mappings must be 1:1 for outbound mappings. For each local attribute name, you can only specify one assertion attribute name.

Attribute name mapping is configured through the Fusion Middleware Control Console. See Section 5.9.2.1, "Configuring Attribute Name Mapping" for details.

5.9.1.1.1 Static Attribute Value

You can map a static attribute value (for example., DeploymentVersion=6.4) from the Oracle Identity Federation configuration to an outgoing assertion.

The feature is implemented with two properties for an existing attribute name mapping definition:

- `from-config`, a boolean
- `attribute-value-fromconfig`

If `from-config` is `true`, and if that attribute needs to be included in an outgoing assertion, the value stored in `attribute-value-fromconfig` is placed in the outgoing assertion.

If `from-config` is `false`, the value set for the attribute in Fusion Middleware Control is used.

The steps involve creating an attribute name mapping definition to the list of attribute mappings, and setting these properties:

- the internal name of the attribute, referenced by `datastore-attr`
- the name of the attribute as it will appear in the assertion, referenced by `assertion-attr`
- the format or namespace of the SAML attribute (which can be blank), referenced by `format-attr`
- a flag referenced by `send-with-sso` indicating whether the attribute should be sent in an SSO assertion
- a flag referenced by `from-session` indicating whether to retrieve the attribute value from the Oracle Identity Federation user session,
- a flag referenced by `from-config` indicating whether the attribute value is static
- a property referenced by `attribute-value-fromconfig` containing the static value

The WLST commands to configure static attribute mapping are as follows:

Note: You must replace the sample host:port, map name, and static values used in this example with valid values.

- Create an attribute name mapping definition for an attribute - with local name set to cn and assertion name set to commonName - for providerid http://myhost.domain.com:7499/fed/sp, if it does not already exist:

```
createFederationPropertyMap('http://myhost.domain.com:7499/fed/sp',
'attributelist')
createFederationPropertyMapInMap('http://myhost.domain.com:7499/fed/sp','attrib
utelist','cncommonName')
```

- Add the new set of properties for the attribute, if they do not already exist, using these guidelines. (These properties are added to the previously created element.)

Note: When creating a mapping from scratch, you must also include the attribute "require-from-infocard".

Property	Set to
datastore-attr	cn
assertion-attr	commonName
format-attr	empty value
send-with-sso	true (always send the attribute with SSO assertions) or false (do not send attribute)
from-session	true (retrieve the attribute value from the Oracle Identity Federation user session) or false (do not retrieve value)
from-config	true (attribute has a static value) or false
attribute-value-fromconfig	static value if needed

For example:

```
addFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist', 'cncommonName', 'datastore-attr', 'cn', 'string')

addFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist', 'cncommonName', 'assertion-attr', 'commonName', 'string')

addFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist', 'cncommonName', 'format-attr', '', 'string')

addFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist', 'cncommonName', 'send-with-sso', 'true', 'boolean')

addFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist', 'cncommonName', 'from-session', 'false', 'boolean')

addFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist', 'cncommonName', 'from-config', 'true', 'boolean')

addFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist', 'cncommonName', 'attribute-value-fromconfig',
'cn=users,dc=oracle,dc=com', 'string')
```

Here is another example that includes the "require-from-infocard" property:

```
addFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist', 'cncommonName', 'require-from-infocard', 'true', 'boolean')
```

- To remove a property and the corresponding map for the attribute:

```
removeFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist','cncommonName','attribute-value-fromconfig')

removeFederationMapEntryInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist','cncommonName','from-config')

removeFederationMapInMap('http://myhost.domain.com:7499/fed/sp',
'attributelist','cncommonName')
```

5.9.1.2 Attribute Value Mapping

Attribute value mapping allows the administrator to specify the value that a local attribute should be assigned in a SAML message when sending or receiving messages.

Attribute value mapping has these characteristics:

- A value mapping consists of a combination, or duet, of a local value and the corresponding external value.
- Value mappings can be defined for any local attributes. Multiple value mappings can be defined for each local attribute.
- Different external values can be mapped to the same local value using value mappings. A default attribute is used to determine which external value will be used in outgoing mode.
- Different local values can be mapped to the same external value by means of value mappings. A default attribute is used to determine which local value to use in incoming mode when mapping external values into local values.

Oracle Identity Federation exercises attribute value mapping when acting as a:

- Attribute Authority
- Attribute Requester
- Identity Provider, when sending attributes in SSO assertions

Attribute value mapping is configured through the Fusion Middleware Control Console. See Section 5.9.2.2, "Configuring Attribute Value Mapping" for details.

5.9.1.3 Attribute Value Filtering

Attribute value filtering allows the administrator to specify which local values are allowed when sending a SAML message.

Attribute value filtering has these characteristics:

- Filter rules can be defined for any local attributes. A filter rule evaluates each attribute value to determine if it can be sent. If the evaluation is positive, the value is sent; otherwise, it is removed from the list of attribute values to be sent.
- Multiple filter rules can be defined for each local attribute. When sending a value, Oracle Identity Federation can be set up to either:
  - send only after all filters evaluate successfully
  - send if at least one filter evaluates successfully
- The administrator defines a filtering rule by specifying the type of comparison, and the string value to compare (see Section 5.9.2.3, "Configuring Attribute Value Filtering").
- Oracle Identity Federation supports these comparison types when comparing the attribute value to a string:
  - equals
  - not equals
  - starts with
  - ends with
  - contains
  - does not contain
  - equals null
  - not equals null
- In addition to these comparison types, filtering supports regular expressions, allowing the user to match the attribute value against a regular expression. See Section 5.9.2.3.1, "Filtering Conditions" in Section 5.9.2.3, "Configuring Attribute Value Filtering" for details.
- The filtering rules allow you to specify whether the comparison will be case-sensitive.

Oracle Identity Federation exercises attribute value filtering when acting as a:

- Attribute Authority
- Identity Provider, when sending attributes in SSO assertions

You configure this feature through the Fusion Middleware Control Console. See Section 5.9.2.3, "Configuring Attribute Value Filtering" for details.

5.9.2 Mapping and Filtering Configuration

This section explains how to configure mapping and filtering:

- Configuring Attribute Name Mapping
- Configuring Attribute Value Mapping
- Configuring Attribute Value Filtering

5.9.2.1 Configuring Attribute Name Mapping

Configuration of attribute name mapping serves these purposes:

On the IdP side:

- mapping attribute names contained in assertions to local attribute names
- determining which local attributes can be sent to the peer provider. Defining an attribute name mapping for a peer provider will authorize Oracle Identity Federation to send this attribute to the remote server.

On the SP side:

- mapping attribute names contained in SOAP client requests to names in attribute queries to the attribute authority

Take these steps to define attribute name mappings:

See Also: Section 5.11, "Configuring Federations"

On the IdP Side

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Federations**.
3. Select the Attribute Requester with which you want to configure attribute sharing, and click **Edit**.
4. Click **Edit Attribute Mappings and Filters**.
5. Under the **Name Mappings** tab, click **Add** to add an attribute name mapping, with the following fields:
   - User Attribute Name: The name of the local attribute in the user repository. If the UserID should be mapped to the assertion attribute, set this to "orafed-userid".
   - Assertion Attribute Name: The name that will be used to identify the attribute in the Attribute Query and assertion
   - Format or Namespace: An optional field used to specify the format or the namespace of the SAML attribute, depending on the version.
     - For SAML 1.x, this field's value is used to set the SAML attribute's namespace.
     - For SAML 2.0, this value is used to set the SAML attribute's NameFormat; if this field is empty, the NameFormat of the SAML attribute will be set to urn:oasis:names:tc:SAML:2.0:attrname-format:basic; otherwise the NameFormat will hold the value specified in this field.
   - Send with SSO Assertions: Indicates whether the attribute should be sent in the assertion during an SSO operation.

Note: In order for the identity provider to send an attribute to a peer provider, a mapping for this attribute must be defined as explained above.

- Get Value from User Session: Indicates whether the attribute value should be obtained from the user session.
- Require from Infocard: Indicates whether the attribute must be passed in from Infocard.

On the SP Side

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Federations**.
3. Select the attribute authority with which you want to configure attribute sharing, and click **Edit**.
4. Select **Update Manually**; under Oracle Identity Federation Settings, click **Edit Attribute Mappings and Filters**.
5. Under the Name Mappings tab, click **Add** to add an attribute name mapping, with the following fields:
   - User Attribute Name: The name used by the SOAP client in the AttributeRequest
   - Assertion Attribute Name: The name that will be used to identify the attribute in the attribute query and assertion
   - Format or Namespace: An optional field used to specify the format or the namespace of the SAML attribute, depending on the version
     - For SAML 1.x, this field's value is used to set the SAML attribute's namespace
     - For SAML 2.0, this value is used to set the SAML attribute's NameFormat; if this field is empty, the NameFormat of the SAML attribute will be set to urn:oasis:names:tc:SAML:2.0:attrname-format:basic; otherwise the NameFormat will hold the value specified in this field
   - Get Value from User Session: Indicates whether the attribute value should be obtained from the user session.
   - Require from Infocard: Indicates whether the attribute must be passed in from Infocard.

Note: If no mapping is found for an attribute name, the service provider will map the name to itself.

Example

The following attribute name configuration will yield the results shown here.

Name Mapping in SP:

User Attribute	Assertion Attribute
phone	telephone
userid	username
email	emailaddress
id	idnumber

Name Mapping in IdP:

Assertion Attribute	User Attribute
lastname	sn
idnumber	employeenumber
telephone	telephonenumber
title	title
username	uid
emailaddress	mail
firstname	givenname

Results:

Attribute in SOAP client Request	Attribute in SAML Attribute Query	Attribute in User Datastore	Attribute in SAML Assertion	Attribute in Response to SOAP client
lastname	lastname	sn	lastname	lastname
id	idnumber	employeenumber	idnumber	id
phone	telephone	telephonenumber	telephone	phone
title	title	title	title	title
userid	username	uid	username	userid
email	emailaddress	mail	emailaddress	email
firstname	firstname	givenname	firstname	firstname
middlename	middlename	-	-	-

Note that:

- For attributes `lastname`, `title`, `firstname`, there is no mapping in the SP, so they are mapped to themselves.
- For attribute `middlename`, there is no mapping in the IdP, so the IdP does not return any values for this attribute. If the attribute name used in the Attribute Query/assertion is the same as in the user data store, you need to explicitly define a mapping for the attribute name that maps the name to itself, as is done here for attribute `title`.

5.9.2.2 Configuring Attribute Value Mapping

Take these steps to define attribute value mappings:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Federations**.
3. Select the peer provider with which you want to configure attribute sharing, and click **Edit**.
4. Select **Update Manually**; under Oracle Identity Federation Settings, click **Edit Attribute Mappings and Filters**.
5. Under the **Value Mappings** tab, click **Add** to add an attribute value mapping, with the following fields:
   - Attribute Name: The name of the local attribute in the user repository
   - Unmapped Values: Check Send to allow Oracle Identity Federation to send values for which a mapping is not defined. Check Receive to allow Oracle Identity Federation to receive values for which a mapping is not defined.
   - A list of Local to External Value Mappings:
     - Local Value: The local value of the attribute
     - External Value: The corresponding value to send in external messages
     - Ignore Case: If checked, indicates that the string comparison should be case-sensitive when matching attribute values.
     - Local Null: If checked, indicates that the local value equals a null string (different from an empty string "").
     - External Null: If checked, indicates that the external value equals a null string (different from an empty string "").
     - Default: If selected, indicates this local value will be used in case an incoming external value can be mapped to several local values.

Example

This value mappings configuration for the attribute `title` will yield the following results:

- Attribute Name: title
- Unmapped Values:
  - Send: checked
  - Receive: checked
- Value Mappings:

Local Value	External Value	Ignore Case	Local Null	External Null	Default
Senior Member of Technical Staff	smts	checked			checked
Principal Member of Technical Staff	pmts	checked			
	None		checked		

Local Value	External Value	Ignore Case	Local Null	External Null	Default
Senior Member of Technical Staff	srmts	checked			
Consulting Member of Technical Staff	cmts	checked			

Results:

External Value	maps to Local Value
Consulting Member of Technical Staff	cmts
PRINCIPAL MEMBER OF TECHNICAL STAFF	pmts
Principal Member of Technical Staff	pmts
Senior Member of Technical Staff	smts
Vice President	Vice President

Local Value	maps to External Value
NULL	None
smts	Senior Member of Technical Staff
srmts	Senior Member of Technical Staff
CEO	CEO

Note that:

- Since we defined value mappings to be case-insensitive, both "PRINCIPAL MEMBER OF TECHNICAL STAFF" and "Principal Member of Technical Staff" get mapped to `pmts`.
- Since Unmapped Values: Send is checked and there is no rule defined for value "Vice President", it is mapped to itself.
- Since we defined `smts` to be the default local value for "Senior Member of Technical Staff", "Senior Member of Technical Staff" gets mapped to `smts` even though `srmts` also maps to "Senior Member of Technical Staff".
- A local value of NULL, gets mapped to the string `None`.
- Both `smts` and `srmts` map to "Senior Member of Technical Staff"
- Since Unmapped Values: **Receive** is checked and there is no rule defined for "CEO", it is mapped to itself.

5.9.2.3 Configuring Attribute Value Filtering

Take these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Federations**.
3. Select the attribute requester with which you want to configure attribute sharing, and click **Edit**.
4. Select **Update Manually**; under Oracle Identity Federation Settings, click **Edit Attribute Mappings and Filters**.
5. Under the **Value Filters** tab, click **Add** to add an attribute value filter, with the following fields:
   - Attribute Name: The name of the local attribute in the user repository
   - Condition Operator: Select "and" to indicates that all conditions need to be met for an attribute to be sent. Select "or" to indicate meeting one condition is enough to send an attribute.
   - A list of filtering rules with the following fields
     - Condition: The condition that will be used to evaluate the attribute value.
     - Expression: The value or regular expression that will be used to evaluate the attribute value.
     - Ignore Case: If checked, indicates that the string comparison should be case-sensitive when matching attribute values.

5.9.2.3.1 Filtering Conditions

Oracle Identity Federation provides several filtering conditions:

- equals: the filtering rule will return true if the expression value is equal to the outgoing attribute value.
- does not equal: the filtering rule will return true if the expression value is different from the outgoing attribute value.
- starts with: the filtering rule will return true if the outgoing attribute value begins with the expression value.
- ends with: the filtering rule will return true if the outgoing attribute value ends with the expression value.
- contains: the filtering rule will return true if the outgoing attribute value contains the expression value.
- does not contain: the filtering rule will return true if the outgoing attribute value does not contain the expression value.
- equals null: the filtering rule will return true if the outgoing attribute value is null.
- does not equal null: the filtering rule will return true if the outgoing attribute value is not null.
- regexp: the filtering rule will return true if the outgoing attribute value matches the regular expression, which is defined in the expression value.

Note: The rules are used to determine the allowed values. Consequently, if a rule evaluates to true, this means that it is permissible to send the value.

When the filtering condition is set to `regexp`, the expression value must be a standard Unix regular expression.

See `http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html` for details about regular expression constructs.

Note: When the filtering condition is set to regexp, the `ignoreCase` flag is disregarded during attribute value processing because regular expressions already support case-insensitivity.

Table 5–8 contains some examples illustrating the use of the `regexp` filtering condition:

Table 5–8 Examples of using the regexp Filtering Condition

Regular Expression	Meaning
.*rector	any string which ends with "rector"
[^abc]	any character except a, b, or c (negation)
user\d	user0, user1, ..., user9
a*b	any string which begins with 0+ "a" characters and ends with a letter b (for example, aaaaab)

5.9.2.3.2 Examples of Value Filters

This section provides some examples of value filter configuration.

Example 1

This value filters configuration for the attribute `title`, will yield the following results:

- Attribute Name: title
- Condition Operator: and
- Value Filters:

Condition	Expression	Ignore Case
does not equal	Vice-President	checked
contains	President	checked

Results:

Value	Send Value?
Vice-President	no
President	yes
Vice-president	no
Senior Vice-President	yes

Value	Send Value?

Example 2

Suppose attribute value mappings are defined as in the example in Section 5.9.2.2, "Configuring Attribute Value Mapping". This value filters configuration for attribute `title`, will yield the following results:

- Attribute Name: title
- Condition Operator: and
- Value Filters:

Condition	Expression	Ignore Case
does not equal	mngr	true
ends with	mts	false

Results:

Value	Send Value?	Value Sent
mngr	no	
cmts	yes	Consulting Member of Technical Staff

Note that:

- For a value to be sent, it must not equal mngr, so the value `mngr` will not be sent.
- `cmts` can be sent (all filter conditions evaluate to true), and it is mapped to "Consulting Member of Technical Staff".
- The same results would apply for the following value filters:

Condition	Expression	Ignore Case
does not equal	mngr	true
regexp	*mts	

5.10 Configuring Security and Trust

You use the security and trust pages to configure keystores and certificates for the Oracle Identity Federation server.

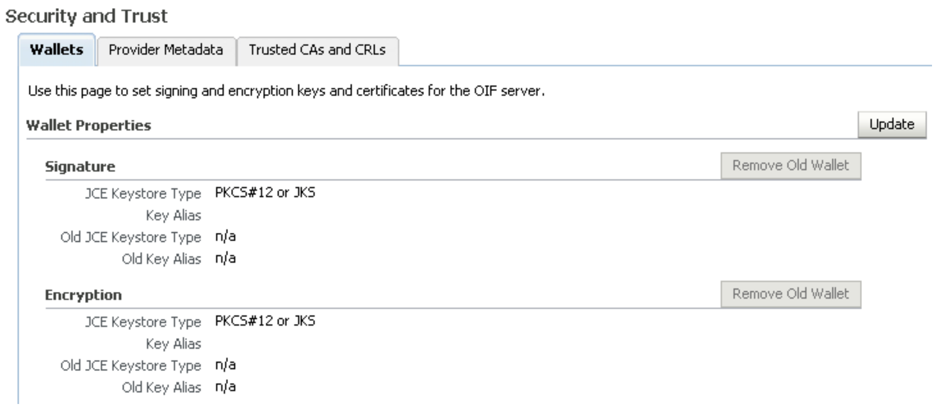
To access these pages, start from the Oracle Identity Federation drop-down adjacent to the Topology icon, and navigate to **Administration**, then **Security and Trust**.

This section contains these topics relating to trust configuration:

- Security and Trust - Wallet
- Security and Trust - Provider Metadata
- Security and Trust - Trusted CAs and CRLs

5.10.1 Security and Trust - Wallet

Signing and encryption certificates for this server instance are stored in wallets. Use this page to manage the signing and encryption wallets.



Use this page to set signing and encryption keys and certificates for the OIF server.

Wallet Properties

Signature

JCE Keystore Type	PKCS#12 or JKS
Key Alias	
Old JCE Keystore Type	n/a
Old Key Alias	n/a

Encryption

JCE Keystore Type	PKCS#12 or JKS
Key Alias	
Old JCE Keystore Type	n/a
Old Key Alias	n/a

The page shows:

- The type of the signature wallet; for example, PKCS#12 or JKS.
- The alias of the signing key in the wallet.
- The type of the previous signature wallet; for example, PKCS#12 or JKS.
- The alias of the previous signing key in previous wallet.
- The type of the encryption wallet; for example, PKCS#12 or JKS.
- The alias of the encryption key in the wallet.
- The type of the previous encryption wallet; for example, PKCS#12 or JKS.
- The alias of the previous encryption key in previous wallet.

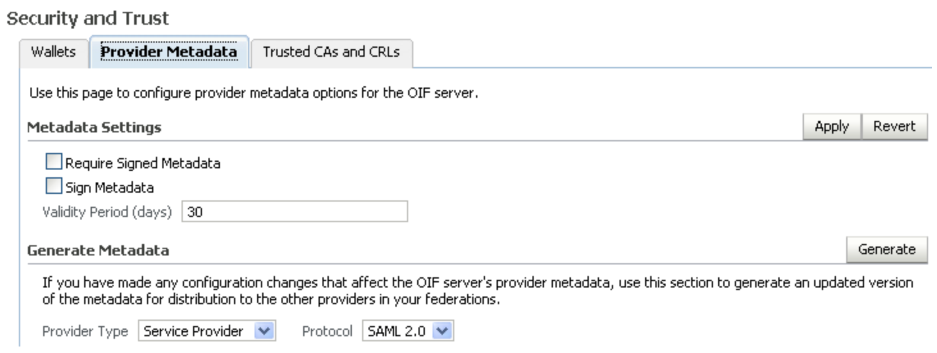
Click **Update** to modify the wallet information. The Update Wallet dialog requires this information for the signing and/or encryption wallet:

- Wallet Location - You can choose an operating system file containing the wallet.
- Password - Enter the password that was used to encrypt the private key.
- Key Password - Only required for JKS and custom Java keystores.
- Signing Key Alias - the alias under which the private key is stored in the wallet.

5.10.2 Security and Trust - Provider Metadata

Use this page to:

- specify metadata signing requirements
- generate updated metadata



Use this page to configure provider metadata options for the OIF server.

Metadata Settings

- [ ] Require Signed Metadata
- [ ] Sign Metadata

Validity Period (days) 30

Generate Metadata

If you have made any configuration changes that affect the OIF server's provider metadata, use this section to generate an updated version of the metadata for distribution to the other providers in your federations.

Provider Type Service Provider   Protocol SAML 2.0

Metadata Signing

Oracle Identity Federation supports XML digital signatures in the XML metadata documents that describe the services published by a compliant federation server. Oracle Identity Federation provides the following support for metadata signatures:

- digitally signing the metadata Oracle Identity Federation publishes
- verifying any XML digital signature present on a metadata document that is being uploaded to the server. If the verification fails, the metadata will not be uploaded.
- configuring the server to require an XML digital signature on provider metadata in order to upload it to the trusted providers.

Use this section of the page to specify metadata signing. Provide the following information:

- Require Signed Metadata - Check the box to specify that Oracle Identity Federation must require signed metadata when importing a descriptor to the trusted providers. Thus, peer providers must provide signed metadata to the server.
- Sign Metadata - Check the box to require the Oracle Identity Federation server to sign its metadata.
- Validity Period - Enter the validity period in days.

Click **Apply** to save the changes, or **Revert** to reset the fields to their previous state.

Generate Metadata

Use this section of the page to generate and distribute metadata to peer providers after making any changes to any server configuration that affects metadata.

- Provider Type - Select the type from the drop-down list.
- Protocol - Select a protocol from the drop-down list.

  SAML 1.0, 1.1, and 2.0 protocols are supported for this function.

> **Note:** Liberty 1.x configuration and metadata uploads are available by using the WLST command-line tool.

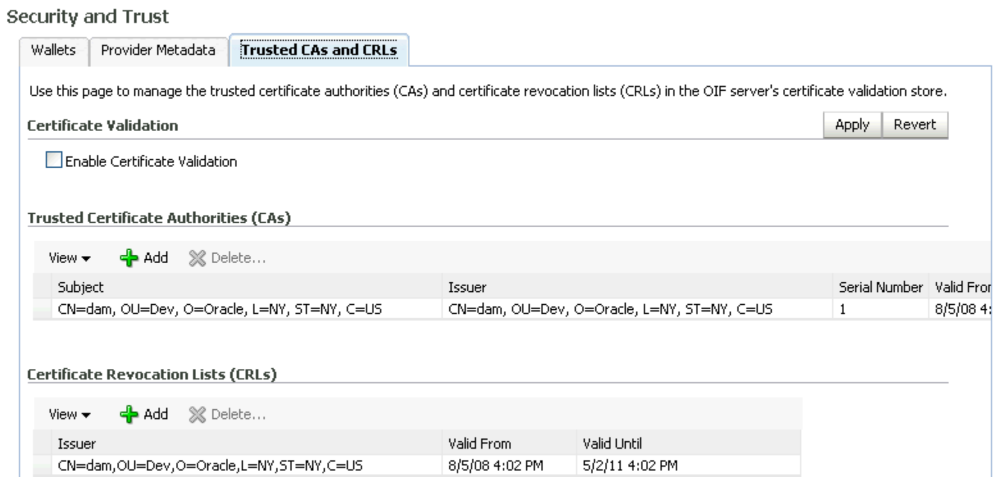
Click **Save** to generate and distribute the metadata.

5.10.3 Security and Trust - Trusted CAs and CRLs

Oracle Identity Federation maintains a credential store to hold trusted certificates and CRLs. When the certificate validation store is enabled, Oracle Identity Federation uses it to validate the certificates needed to verify the signatures on incoming SAML/WS-Federation messages.

Use this page to maintain the following objects in the certificate validation store:

- Certificate Authority (CA) certificates
- Certificate Revocation Lists (CRLs)



Use this page to manage the trusted certificate authorities (CAs) and certificate revocation lists (CRLs) in the OIF server's certificate validation store.

Certificate Validation

☐ Enable Certificate Validation

Trusted Certificate Authorities (CAs)

Subject	Issuer	Serial Number	Valid Fror
CN=dam, OU=Dev, O=Oracle, L=NY, ST=NY, C=US	CN=dam, OU=Dev, O=Oracle, L=NY, ST=NY, C=US	1	8/5/08 4:

Certificate Revocation Lists (CRLs)

Issuer	Valid From	Valid Until
CN=dam,OU=Dev,O=Oracle,L=NY,ST=NY,C=US	8/5/08 4:02 PM	5/2/11 4:02 PM

Provide the following information:

- Enable Certificate Validation - Check the box to enable the server to validate certificates.

  Click **Apply** to save the changes, or **Revert** to reset the field to its previous state.

- Trusted Certificate Authorities - The table displays details of CAs trusted by Oracle Identity Federation.

  The CA fields are:

  - **Subject** - this is the CA certificate subject
  - **Issuer** - this is the certificate issuer
  - **Serial Number** - this is the certificate's serial number
  - **Valid From** - this is the start time of the certificate validity period
  - **Valid Until** - this is the end time of the certificate validity period

  Select a CA and click **Delete** to remove it from the store. Click **Add** to add a new trusted CA to the store.

- Certificate Revocation Lists - The CRL table shows a list of Certificate Revocation Lists (CRLs) known to Oracle Identity Federation.

  The CRL fields are:

  - **Issuer** - this is the CA that issued the CRL
  - **Valid From** - this is the start time of the CRL validity period
  - **Valid Until** - this is the end time of the CRL validity period

  Select a CRL and click **Delete** to remove it from the store. Click **Add** to add a new CRL to the store.

5.11 Configuring Federations

See Section 4.3, "Managing Identity Federations".

5.12 Configuring Identities

See Section 4.4, "Configuring Identities".

5.13 Managing Data Stores

This section explains how to configure and manage the different data stores used by Oracle Identity Federation:

- Manage the User Data Store
- Manage the Federation Data Store
- Manage the Session Data Store and the Message Data Store
- Manage the Configuration Data Store
- Create the Oracle Identity Federation Schema Using RCU

5.13.1 Manage the User Data Store

This section explains how to configure user data stores for Oracle Identity Federation:

- Configuring Oracle Identity Federation for RDBMS User Data Store
- Configuring Oracle Identity Federation for an LDAP User Data Store
- Configuring Oracle Virtual Directory as User Data Store
- Configuring a Redundancy User Data Store
- Configuring No User Data Store

5.13.1.1 Configuring Oracle Identity Federation for RDBMS User Data Store

In order for Oracle Identity Federation to use a database as the user data store, this database must have a table, referred to as the user table, that contains user information.   The user table must have a column that contains the User ID with which the user will be identified in Oracle Identity Federation. The User ID must always be present and must be unique across all users.   If Attribute Sharing or User Mapping with Attributes will be used, columns for these attributes must also be present in the user table.

To configure Oracle Identity Federation to use an RDBMS user data store:

1. Create a JDBC Data Source
2. Modify Oracle Identity Federation Data Store Configuration

Create a JDBC Data Source

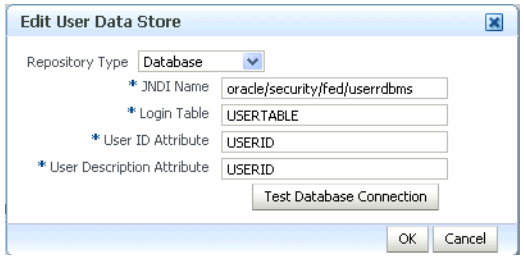
Follow these steps to create a JDBC data source:

> **See Also:** Getting Started with Oracle WebLogic Server Administration Console in the *Oracle Fusion Middleware Administrator's Guide*.

1. Log in to the WebLogic Administration Console.

2. Navigate to **Services**, then **JDBC**, then **Data Sources**.
3. Click **New**.
4. Choose a name and a JNDI name for the new data source, and enter the database information. Choose the WebLogic managed server where Oracle Identity Federation is deployed as the target of this data source.

Configure an RDBMS User Data Store



Repository Type	Database
* JNDI Name	oracle/security/fed/userrdbms
* Login Table	USERTABLE
* User ID Attribute	USERID
* User Description Attribute	USERID

Follow these steps to configure an RDBMS user data store:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance
2. Navigate to **Administration**, then **Data Stores**.
3. In the User Data Store section, click **Edit**.
4. Select **Database** from the **Repository Type** dropdown list.
5. Enter the following properties:
   - JNDI Name: The JNDI of the data source created in the WebLogic Administration Console.
   - Login Table: The name of the user table.
   - User ID Attribute: The name of the User ID column in the user table.
   - User Description Attribute: The name of the User Description column in the user table.
6. Click **OK**.

Example

Consider the following user table, named "UserInformation", and suppose the JNDI name of the data source is "MyCorpUserDS".

Username	FirstName	LastName	FullName	Email
alice	Alice	Smith	Alice Smith	alice@mycorp.com
bob	Robert	Jones	Robert Jones	bob@mycorp.com
charlie	Charles	Johnson	Charles Johnson	charlie@mycorp.com
david	David	Jones	David Jones	david@mycorp.com
robert	Robert	Williams	Robert Williams	williams@mycorp.com

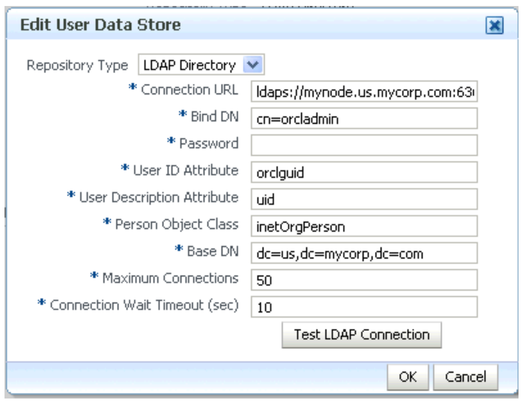
Oracle Identity Federation configuration for the user data store might look like this:

- **JNDI Name**: MyCorpUserDS
- **Login Table**: UserInformation
- **User ID Attribute**: Username
- **User Description Attribute**: Full Name

Alternatively, the configuration could be:

- **JNDI Name**: MyCorpUserDS
- **Login Table**: UserInformation
- **User ID Attribute**: Username
- **User Description Attribute**: Username

5.13.1.2 Configuring Oracle Identity Federation for an LDAP User Data Store


Edit User Data Store

Repository Type	LDAP Directory
* Connection URL	ldaps://mynode.us.mycorp.com:63
* Bind DN	cn=orcladmin
* Password	
* User ID Attribute	orclguid
* User Description Attribute	uid
* Person Object Class	inetOrgPerson
* Base DN	dc=us,dc=mycorp,dc=com
* Maximum Connections	50
* Connection Wait Timeout (sec)	10

Follow these steps to configure an LDAP user data store:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Data Stores**.
3. In the **User Data Store** section, click **Edit**.
4. Select **LDAP Directory** from the **Repository Type** dropdown list.

The fields to set up this configuration are as follows:

- Connection URL - This is the LDAP URL to connect to the server. For example, ldap://ldap.oif.com:389.
- Bind DN - This is the administrator account DN to use to connect to the LDAP server. For example, cn=orcladmin
- Password - Administrator password to connect to LDAP server
- UserID attribute - This is the LDAP attribute used to identify user during authentication, for example uid. Here are examples of the User ID attribute for different types of directory servers:
  - Oracle Internet Directory: uid
  - Sun Java System Directory Server: uid
  - Microsoft Active Directory: sAMAccountName

- User Description attribute - This is the human-readable LDAP attribute used to identify the owner of a federation record, for example uid. Here are examples of the User Description Attribute for different types of directory servers:
  - Oracle Internet Directory: uid
  - Sun Java System Directory Server: uid
  - Microsoft Active Directory: sAMAccountName
- Person Object Class - Object classes define what data or attributes are associated with an object. A person object class refers to the attributes of a "person" object; in our context, it is the owner of a federated identity. A directory may utilize one or more object classes to hold person data (names, addresses, and so on). Enter the LDAP object class representing an LDAP user entry in the server. Here are examples of the person object class for different types of directory servers:
  - Oracle Internet Directory: inetOrgPerson
  - Sun Java System Directory Server: inetOrgPerson
  - Microsoft Active Directory: user
- Base DN - This is the directory to which the search for users should be confined.
- Maximum Connections - This is the maximum number of LDAP connections that Oracle Identity Federation will simultaneously open to the LDAP server.
- Connection Wait Timeout - This is the timeout, in minutes, to use when Oracle Identity Federation opens a connection to the LDAP server.

5.13.1.3 Configuring Oracle Virtual Directory as User Data Store

Oracle Identity Federation can be integrated with Oracle Virtual Directory; when using Oracle Virtual Directory as the user data store, ensure that the base DN, person object class, unique user id and user description attribute settings are valid for all directory structures connected to Oracle Virtual Directory.

5.13.1.4 Configuring a Redundancy User Data Store

Redundancy is supported for the user data stores; this section explains how to set up redundancy user data stores.

There are two ways to set up redundancy user data stores:

1. In the user data store configuration, in the **Server URL** field, enter a list of space-separated ldap URLs.

   For example:

   ```
   ldap://ldap1.oif.mycorp.com ldap://ldap2.oif.mycorp.com
   ldap://ldap3.oif.mycorp.com
   ```

   or

2. Set up a load balancer in front of the LDAP servers and set the "ldaphaenabled" property in Oracle Identity Federation configuration to `true`. For details about this task, see Section 6.4.1, "Configuring High Availability LDAP Servers".

5.13.1.5 Configuring No User Data Store

You can configure Oracle Identity Federation not to use a user data store at runtime. In this configuration, the only user information available to the server is the user identifier:

- after local authentication, or
- after the incoming assertion is mapped with the use of a federated identity record, when the server acts as a Service Provider

To configure Oracle Identity Federation not to use a user data store:

1. Modify Oracle Identity Federation Data Store Configuration
2. Modify Oracle Identity Federation Configuration to use the user identifier

Modify Oracle Identity Federation Data Store Configuration

Follow these steps to configure no user data store:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Data Stores**.
3. In the User Data Store section, click **Edit**.
4. Select `None` from the Repository Type dropdown list.
5. Click **OK**.

Modify Oracle Identity Federation Configuration to use the User Identifier

When `None` is selected as the user data store, you can configure Oracle Identity Federation so that the user identifier will be used to populate assertion data, or to configure the federation data store.

If Oracle Identity Federation acts as an Identity Provider, you can configure the server to:

- set the user identifier as the Assertion Name ID

  To achieve this, navigate to the NameID format table, and set the user attribute for the NameID format to `orafed-userid`.

- add the user identifier as an assertion attribute

  To achieve this, navigate to the configuration screen for the remote Service Provider to which the assertion will be sent, define an attribute to be sent, and set the user attribute to `orafed-userid`.

If a federation data store is in use, be sure to configure Oracle Identity Federation to use `orafed-userid` as the user ID attribute and user description attribute in the section that configures the federation data store.

5.13.2 Manage the Federation Data Store

Oracle Identity Federation provides the option of configuring a back-end data store to store records containing federated identity information.

If configured to use a federation data store of type XML, LDAP, or RDBMS, Oracle Identity Federation will create a federation record for each user, store this record in the selected data store, and use it in Single Sign-On to create an assertion (if acting as the identity provider), or to map the assertion received from the IdP to a user (if acting as the service provider).

To use persistent Name IDs with the SAML 2.0 protocol requires a Federation Datastore, as an opaque identifier must be created for each user (that is, the Name ID used to identify the user cannot be an attribute from the user datastore, and must thus be created and stored separately).

Oracle Identity Federation will create the federation record on the first time that the user performs a Single Sign-On operation. If acting as a service provider and using persistent Name IDs, since the Name ID does not contain any user information, Oracle Identity Federation will prompt the user for local authentication and create a federation record taking user information from this authentication. Once the federation has been created, Oracle Identity Federation (acting as a service provider) will not ask the user to login locally, since it can automatically map the opaque Name ID in the assertion to the user, using the federation record.

If the Federation Datastore is set to `None`, then Oracle Identity Federation will not create, store, or use federation records, but rather use attributes in the user data store to identify users, either to create assertions, in the case when it is acting as the identity provider, or to map assertions to users, in the case when it is acting as the service provider.

Note: Configuring XML as the federation store is not recommended for production environments. Use an RDBMS or LDAP store in production environments.

Note: You can also set up redundancy federation data stores. For more information see Section 5.13.1.4, "Configuring a Redundancy User Data Store".

Guidelines

Here are some general guidelines:

- If the Federation Datastore is set to `None`:
  - Persistent Name IDs *cannot* be used
  - If acting as an identity provider, Oracle Identity Federation will create an assertion by mapping the Name ID format to be used to an attribute in the user data store, and using the value of this attribute as the Name ID.
  - If acting as a service provider. Oracle Identity Federation will map the assertion received from the identity provider by either using an attribute query, or by mapping the Name ID in the assertion to an entry in the user data store.
- If the Federation Datastore is set to XML, LDAP or RDBMS:
  - Persistent Name IDs *can* be used
  - If acting as an identity provider, Oracle Identity Federation will use the Name ID stored in the federation record created for the user, when creating the assertion.
  - If acting as a service provider, Oracle Identity Federation will map the assertion to a user by finding the federation record with the Name ID included in the assertion.
  - If acting as a service provider and using persistent Name IDs, Oracle Identity Federation will prompt for local authentication the first time SSO is performed with a given user and a given provider.

Subsequent sections cover these topics:

- Configuring Oracle Identity Federation for an RDMBS Federation Data Store

- Configuring Oracle Identity Federation for an LDAP Federation Data Store
- Configuring Oracle Identity Federation for an XML Federation Data Store
- Configuring Oracle Virtual Directory as Federation Data Store

5.13.2.1 Configuring Oracle Identity Federation for an RDMBS Federation Data Store

The high-level steps to configure an RDBMS federation data store are:

1. Create a JDBC data source.
2. Run RCU to create the Oracle Identity Federation schema.

> **Note:** Be sure to write down the Oracle Identity Federation schema owner and password that is shown in RCU. It is of the form *`PREFIX_`* `OIF`; you will need to provide this information when configuring Oracle Identity Federation.

3. Modify Oracle Identity Federation data store configuration.

   This involves configuring Oracle Identity Federation to use the new data source from Step 1, and configuring the federation data store.

We will now describe each step in detail.

Create a JDBC Data Source

Follow these steps to create a JDBC Data Source:

> **See Also:** Getting Started with Oracle WebLogic Server Administration Console in the *Oracle Fusion Middleware Administrator's Guide*.

1. Log in to the WebLogic Administration Console.
2. Navigate to **Services**, then **JDBC**, then **Data Sources**.
3. Click **New**.
4. Choose a Name and a JNDI Name for the new data source, and enter the database information. Choose the WebLogic managed server where Oracle Identity Federation is deployed as the target of this data source.

Create Oracle Identity Federation Schema

Follow the steps described in Section 5.13.5, "Create the Oracle Identity Federation Schema Using RCU" to create the Oracle Identity Federation schema.

Modify Oracle Identity Federation Data Store Configuration

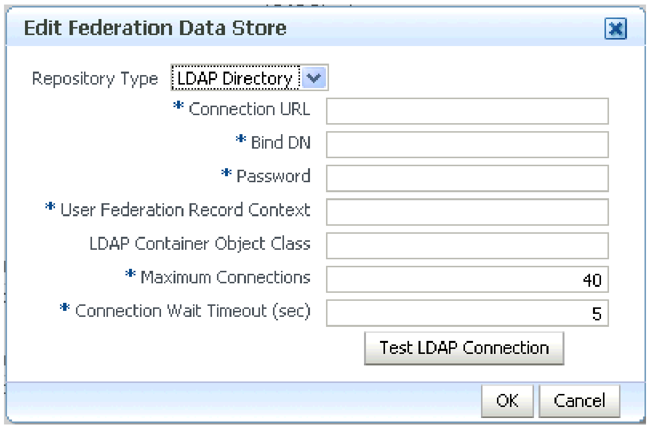
1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Data Stores**.
3. In the **Federation Data Store** section, click **Edit**.
4. Select **Database** from the **Repository Type** dropdown list.

5. Enter the JNDI Name; use the JNDI of the data source created in the WebLogic Administration Console.
6. Click **OK**.

5.13.2.2 Configuring Oracle Identity Federation for an LDAP Federation Data Store

See Also: A Note About the LDAP Schema in Section 2.4.1, "Federation Data Store"

Follow these steps to configure an LDAP federation data store:



Edit Federation Data Store

Repository Type	LDAP Directory
* Connection URL	
* Bind DN	
* Password	
* User Federation Record Context	
LDAP Container Object Class	
* Maximum Connections	40
* Connection Wait Timeout (sec)	5

Connection URL

This is the LDAP URL to connect to the server. For example,

```
ldap://ldap.oif.com:389
```

A space-separated list of LDAP connection URLs can be entered in this field.

Bind DN

This is the administrator account DN to use to connect to the LDAP server. For example,

```
cn=orcladmin
```

User Federation Record Context

This is the LDAP container entry under which all the federation records will be stored. For example,

```
cn=fed,dc=us,dc=oracle,dc=com
```

Note:

- The user federation record context must be compatible with the LDAP container object class, as explained in the description of that field.
- If the user federation record context container object does not exist in the LDAP directory, Oracle Identity Federation will create it at runtime the first time it needs to store a federation record.

LDAP Container Object Class

This is the type of the User Federation Record Context class that Oracle Identity Federation should use when creating the LDAP container, if one does not already exist. If this field is empty, its value will be set to `applicationProcess`.

For Microsoft Active Directory, this field has to be set (to container for example) depending on the user federation record context since applicationProcess will not work under Microsoft Active Directory.

To see how these fields are related, note that the user federation record context references the LDAP container entry under which federation records will be stored, and the LDAP container object class defines the LDAP container attribute used in the DN. In the user federation record context, you specify the DN of the container where the federation records will be stored. That DN contains the parent of an already existing container, and an attribute of the federation record context that is part of its object class.

For example, if the container parent is `dc=us,dc=oracle,dc=com` and the record context attribute is `cn=orclfed`, the requirement that cn must be an attribute of the object class set in the LDAP container object class field (or the applicationProcess object class if not set) ultimately produces a DN such as:

```
cn=orclfed,dc=us,dc=oracle,dc=com
```

If you choose to express the DN of the Federation Record Context as `ou=fed,dc=us,dc=oracle,dc=com`, you will need to set the **LDAP Container Object Class** to an object class that has `ou` as an attribute, like `applicationProcess`.

And if the DN is:

```
cn=fed,dc=us,dc=oracle,dc=com
```

then the **LDAP Container Object Class** must define the `cn` attribute.

Here are examples of the LDAP Container Object Class for different types of directory servers:

- Oracle Internet Directory: `empty`
- Sun Java System Directory Server: `empty`
- Microsoft Active Directory: `container`

Maximum Connections

This is the maximum number of LDAP connections that Oracle Identity Federation will simultaneously open to the LDAP server.

Connection Wait Timeout

This is the timeout, in minutes, to use when Oracle Identity Federation opens a connection to the LDAP server.

5.13.2.3 Configuring Oracle Identity Federation for an XML Federation Data Store

Follow these steps to configure Oracle Identity Federation to use an XML file as the federation data sore.

Note: Configuring XML as the federation store is not recommended in production environments. Use an RDBMS or LDAP store in production environments.

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Data Stores**.
3. In the **Federation Data Store** section, click **Edit**.
4. Select XML file from the **Repository Type** dropdown list.
5. Click **OK**.

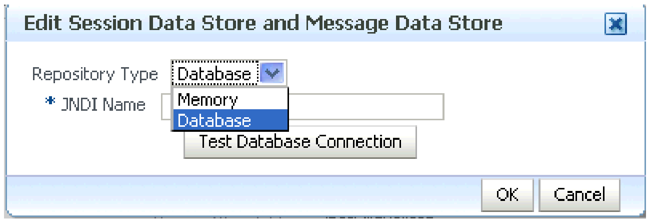
5.13.2.4 Configuring Oracle Virtual Directory as Federation Data Store

When integrating Oracle Virtual Directory into the Oracle Identity Federation environment to serve as the federation data store, ensure that the fed record context and the LDAP container object class settings are valid for the particular directory structure used to store federation records; that is, they must be valid for the directory structure of the LDAP Server that is "referenced" by the federation record context.

5.13.3 Manage the Session Data Store and the Message Data Store

Oracle Identity Federation uses the message data store and the user session data store for storing transient data like federation protocol/session state. The message data store together with the user session data store is also referred to as the transient data store.

Transient data can be stored either in memory or in a relational database.


Edit Session Data Store and Message Data Store

Repository Type: Database

* JNDI Name

Follow these steps to configure Oracle Identity Federation to use an in-memory session/message data store:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Data Stores**.
3. In the **Session Data Store and Message Data Store** section, click **Edit**.
4. Select **Memory** from the **Repository Type** dropdown list.

To configure Oracle Identity Federation to use an RDBMS session/message data store, the high-level steps are:

1. Create a JDBC Data Source.
2. Run RCU to create the Oracle Identity Federation schema.

Note: Be sure to write down the Oracle Identity Federation schema owner and password that is shown in RCU. It is of the form *PREFIX*_OIF; you will need to provide this information when configuring Oracle Identity Federation.

---

3. Modify the Oracle Identity Federation data store configuration.

This involves configuring Oracle Identity Federation to use the new data source from Step 1, and configuring the federation data store.

We will now describe each step in detail.

Create a JDBC Data Source

Follow these steps to create a JDBC data source:

> **See Also:** Getting Started with Oracle WebLogic Server Administration Console in the *Oracle Fusion Middleware Administrator's Guide*.

1. Log in to the WebLogic Administration Console.
2. Navigate to **Services**, then **JDBC**, then **Data Sources**.
3. Click **New**.
4. Choose a Name and a JNDI Name for the new data source, and enter the database information. Choose the WebLogic managed server where Oracle Identity Federation is deployed as the target of this data source.

Create Oracle Identity Federation Schema

Follow the steps described in Section 5.13.5, "Create the Oracle Identity Federation Schema Using RCU" to create the Oracle Identity Federation schema.

Modify Oracle Identity Federation Data Store Configuration

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Data Stores**.
3. In the **Configuration Data Store** section, click **Edit**.
4. Select **Database** from the **Repository Type** dropdown list.
5. Enter the JNDI Name; use the JNDI of the data source created in the WebLogic Administration Console.
6. Click **OK**.

5.13.4 Manage the Configuration Data Store

Oracle Identity Federation uses the configuration data store to store its configuration artifacts. The configuration store can either be an XML file or a relational database. If your deployment is a High Availability deployment, you must use a relational database as the configuration data store.

This section contains these topics:

- Using a File System Configuration Data Store
- Using an RDBMS Configuration Data Store
- When the RDBMS Configuration Data Store is Down

5.13.4.1 Using a File System Configuration Data Store

The configuration data is stored in the file system by default during a basic install. To change the store from database to file system:

1. Navigate to **Administration**, then **Data Stores**.

2. In the **Configuration Data Store** section, click **Edit**.
3. Select **File System** from the **Repository Type** dropdown list.

5.13.4.2 Using an RDBMS Configuration Data Store

To configure Oracle Identity Federation to use an RDBMS configuration data store, the high-level steps are:

1. Create a JDBC data source.
2. Run RCU to create the Oracle Identity Federation schema.

   Note: Be sure to write down the Oracle Identity Federation schema owner and password that is shown in RCU. It is of the form *`PREFIX_`*`OIF`; you will need to provide this information when configuring Oracle Identity Federation.

3. Modify Oracle Identity Federation data store configuration.

   This involves configuring Oracle Identity Federation to use the new data source from Step 1, and setting up the configuration data store.

We will now describe each step in detail.

Create a JDBC Data Source

Follow these steps to create a JDBC data source:

> **See Also:** Getting Started with Oracle WebLogic Server Administration Console in the *Oracle Fusion Middleware Administrator's Guide*.

1. Log in to the WebLogic Administration Console.
2. Navigate to **Services**, then **JDBC**, then **Data Sources**.
3. Click **New**.
4. Choose a Name and a JNDI Name for the new data source, and enter the database information. Choose the WebLogic managed server where Oracle Identity Federation is deployed as the target of this data source.

Create Oracle Identity Federation Schema

Follow the steps described in Section 5.13.5, "Create the Oracle Identity Federation Schema Using RCU" to create the Oracle Identity Federation schema.

Modify Oracle Identity Federation Data Store Configuration

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Data Stores**.
3. In the **Configuration Data Store** section, click **Edit**.
4. Select **Database** from the **Repository Type** dropdown list.

5. Enter the JNDI Name; use the JNDI of the data source created in the WebLogic Administration Console.
6. Click **OK**.

5.13.4.3 When the RDBMS Configuration Data Store is Down

If the configuration data store is integrated with an RDBMS, and if the database is down, Oracle Identity Federation can rely on the latest version of configuration data retrieved from the RDBMS, and runtime operations are not affected; nevertheless, you should not perform any configuration while the RDBMS is down, since the changes are not saved in the RDBMS, and so the configuration changes are not propagated to the database.

5.13.5 Create the Oracle Identity Federation Schema Using RCU

This section describes how to create an Oracle Identity Federation schema using RCU. You must create the schemas before data stores can be configured to use a database.

1. Install RCU from the install CD or installer binaries.
2. Run `$RCU_HOME/bin/rcu`.

Note: Be sure to write down the Oracle Identity Federation schema owner and password that is shown in RCU. It is of the form *`PREFIX_`*`OIF`; you will need to provide this information when configuring Oracle Identity Federation.

3. Select **Create** to create components in the database.
4. Enter database connection details in the next screen.
5. Select Oracle Identity Federation from Identity Management from the Select Component screen.
6. Enter the password in the Schema Passwords screen.
7. Proceed to finish the schema creation for Oracle Identity Federation.

5.14 Configuring Authentication Mechanisms

Authentication mechanisms contain the rules that specify how to use an entity's credentials to verify its identity.

Use these sections to learn about and configure authentication mechanisms for server protocols:

- About Authentication Mechanisms
- Configure Authentication Mechanisms - Local
- Configure Authentication Mechanisms - SAML 2.0
- Configure Authentication Mechanisms - SAML 1.x
- Configure Authentication Mechanisms - WS-Federation 1.1

5.14.1 About Authentication Mechanisms

Authentication mechanisms specify the way a user should be challenged when authentication is required; options include username/password, kerberos, and others.

A service provider can request that the identity provider challenge the user in a certain way by specifying an authentication method in its authentication request.

Because the SP and the IdP can communicate using different protocols, Oracle Identity Federation defines local authentication mechanisms to which the protocol-specific methods can be mapped.

For example, both the SAML 2.0 method `urn:oasis:names:tc:SAML:2.0:ac:classes:Password` and the SAML 1.x method `urn:oasis:names:tc:SAML:1.0:am:password` can be mapped to the local authentication mechanism `oracle:fed:authentication:password`.

Oracle Identity Federation will use these local authentication mechanisms in the following situations:

1. The SP can specify in its request an authentication method that describes the way the user should be authenticated. When the Oracle Identity Federation IdP receives this request, it maps the requested method to a local authentication mechanism. If no authentication method was requested, the IdP uses the default authentication mechanism. This authentication mechanism is then mapped to an authentication engine, which determines the way the user is challenged.
2. An SP engine can specify the authentication method the SP will request from the identity provider by specifying a local authentication mechanism. This local mechanism (or the default mechanism if the SP engine did not specify one) is mapped to a protocol-specific method, which is included in the authentication request that the SP sends to the identity provider.
3. If an SP engine does not specify an identity provider to which to send the authentication request, the SP locates the IdP by mapping the authentication mechanism received from the SP engine (or using the default mechanism if the engine did not send a mechanism) to an identity provider.
4. When creating an assertion, the identity provider determines the mechanism used to authenticate the user and specifies the corresponding protocol-specific authentication method in the assertion.
5. When creating a user session, Oracle Identity Federation records the local authentication mechanism used to authenticate the user.
6. When an Oracle Identity Federation IdP uses the Federation SSO Proxy authentication engine, it uses the requested authentication mechanism (or the default mechanism if no method was requested by the SP) to locate the identity provider to which to send the request. See Section 5.15.9.1, "About the Federated SSO Proxy Authentication Engine" for a description of this authentication engine.

Additional topics in this section include:

- Setting the Default Authentication Mechanism
- Mapping from Protocol-specific Methods to Local Mechanisms To Authentication Engines
- Mapping Local Authentication Mechanisms to Identity Providers

5.14.1.1 Setting the Default Authentication Mechanism

If a service provider does not specify an authentication method in its request, the Oracle Identity Federation IdP uses the default authentication mechanism in the cases described earlier.

See Also: Section 5.14.1, "About Authentication Mechanisms"

Follow these steps to set the default authentication mechanism:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Authentication Mechanisms**.
3. Select the default authentication mechanism and click **Apply**.

5.14.1.2 Mapping from Protocol-specific Methods to Local Mechanisms To Authentication Engines

As mentioned earlier, Oracle Identity Federation provides the ability to map:

- protocol specific authentication methods to local authentication mechanisms, and
- local authentication mechanisms to authentication engines

Thus, different authentication engines can be used depending on the authentication method specified by the service provider in its request.

For example, you can define the following mappings for the SAML 2.0 protocol:

```
urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos ->
oracle:fed:authentication:kerberos

oracle:fed:authentication: kerberos -> Custom Kerberos Authentication Engine
```

and:

```
urn:oasis:names:tc:SAML:2.0:ac:classes:Password ->
oracle:fed:authentication:password

oracle:fed:authentication:password  ->  Oracle Single Sign-On
```

If a SAML 2.0 SP requests that the user be authenticated with mechanism `urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos`, Oracle Identity Federation uses the custom authentication engine created to authenticate the user through Kerberos. But if the SP requests the `urn:oasis:names:tc:SAML:2.0:ac:classes:Password` mechanism, the user is authenticated with the Oracle Single Sign-On engine.

To configure:

- the local authentication mechanism to authentication engine mappings and
- protocol-specific authentication method to local authentication mechanism mappings

follow these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Authentication Mechanisms**.
3. In the **Local Mechanisms** tab, enable and map the local mechanisms to authentication engines and click **Apply**.

   For example, add the mappings:

   ```
   oracle:fed:authentication: kerberos -> Custom Kerberos Authentication Engine

   oracle:fed:authentication:password  ->  Oracle Single Sign-On
   ```

4. In each of the protocol-specific tabs (**SAML 2.0**, **SAML 1.x**, **WS-Federation 1.1**), map protocol-specific authentication methods to local mechanisms and click **Apply**.

   For example, in the **SAML 2.0** tab, add the mappings:

   ```
   urn:oasis:names:tc:SAML:2.0:ac:classes: Kerberos -> oracle:fed:authentication:
   kerberos

   urn:oasis:names:tc:SAML:2.0:ac:classes:Password ->
   oracle:fed:authentication:password
   ```

5.14.1.3 Mapping Local Authentication Mechanisms to Identity Providers

Follow these steps to configure mappings from local authentication mechanisms to identity providers:

See Also: Section 5.5, "Configuring Service Providers"

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Service Provider**.
3. In **Protocol Settings**, click **Configure SSO Authentication Mechanism to Identity Provider Mapping**.
4. Click **Add**, and select the authentication mechanism and the identity provider to which it will be mapped.
5. When you are finished adding mappings, click **OK**. Then click **Apply**.

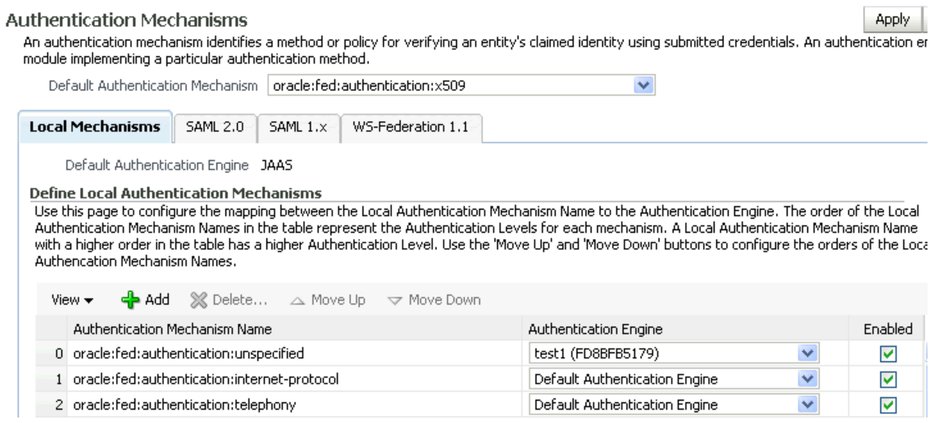
The Federation SSO Proxy authentication engine uses these mappings to determine the identity provider to which to send the request.

An Oracle Identity Federation service provider also uses these mappings to locate the identity provider to which to send the request, when the SP engine does not specify the target IdP in its request to the SP. For example, if using the Oracle Access Manager SP Engine, an Oracle Access Manager scheme can be mapped to an Oracle Identity Federation local authentication mechanism, which is then used to locate the identity provider to which to send the authentication request.

5.14.2 Configure Authentication Mechanisms - Local

Use this page to:

- view, add or delete local authentication mechanisms
- enable or disable selected authentication mechanisms
- map local authentication mechanisms to authentication engines
- update the relative order or the local authentication mechanisms. A mechanism with a higher order in the table has a higher authentication level.



Authentication Mechanisms

Apply

An authentication mechanism identifies a method or policy for verifying an entity's claimed identity using submitted credentials. An authentication er
module implementing a particular authentication method.

Default Authentication Mechanism: oracle:fed:authentication:x509

Default Authentication Engine: JAAS

Define Local Authentication Mechanisms

Use this page to configure the mapping between the Local Authentication Mechanism Name to the Authentication Engine. The order of the Local Authentication Mechanism Names in the table represent the Authentication Levels for each mechanism. A Local Authentication Mechanism Name with a higher order in the table has a higher Authentication Level. Use the 'Move Up' and 'Move Down' buttons to configure the orders of the Loca
Authencation Mechanism Names.

	Authentication Mechanism Name	Authentication Engine	Enabled
0	oracle:fed:authentication:unspecified	test1 (FD8BFB5179)	✓
1	oracle:fed:authentication:internet-protocol	Default Authentication Engine	✓
2	oracle:fed:authentication:telephony	Default Authentication Engine	✓

The authentication mechanisms table displays these columns:

- the name of the local authentication mechanism
- the authentication engine currently associated with a mechanism
- a check-box indicating whether the mechanism is currently active.

To enable (disable) a mechanism, check (uncheck) the corresponding box in the Enable column.

To change the displayed data, use the View drop-down to select the desired fields.

To add a new authentication mechanism, click **Add**.

To delete an existing authentication mechanism, select the row for that mechanism and click Delete. You will be asked to confirm the deletion.

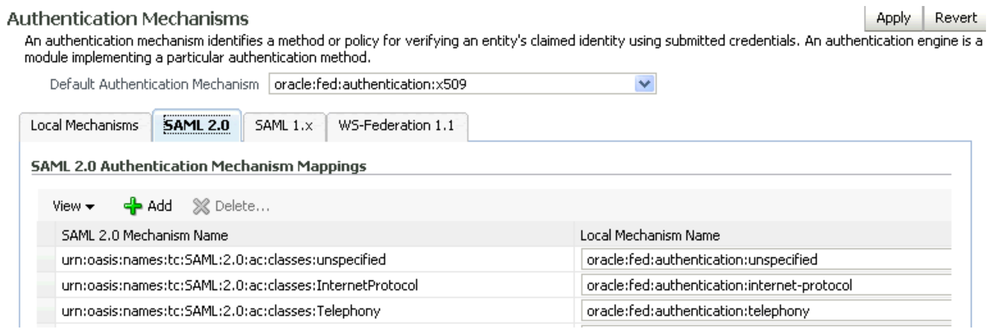
To modify the order of the mechanisms, use the arrow keys at the top of the table after selecting a row.

Click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.14.3 Configure Authentication Mechanisms - SAML 2.0

Use this page to:

- view, add or delete SAML 2.0 authentication mechanisms
- map SAML 2.0 authentication mechanisms to local authentication mechanisms



Authentication Mechanisms

An authentication mechanism identifies a method or policy for verifying an entity's claimed identity using submitted credentials. An authentication engine is a module implementing a particular authentication method.

Default Authentication Mechanism: oracle:fed:authentication:x509

SAML 2.0 Authentication Mechanism Mappings

SAML 2.0 Mechanism Name	Local Mechanism Name
urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified	oracle:fed:authentication:unspecified
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol	oracle:fed:authentication:internet-protocol
urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony	oracle:fed:authentication:telephony

The authentication mechanisms table displays these columns:

- the name of the local authentication mechanism
- the authentication engine currently associated with a mechanism

To change the displayed data, use the View drop-down to select the desired fields.

To add a new authentication mechanism, click **Add**.

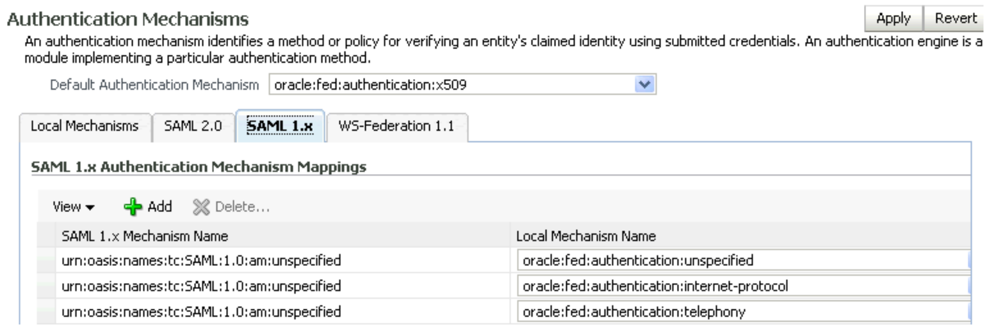
To delete an existing authentication mechanism, select the row for that mechanism and click **Delete**. You will be asked to confirm the deletion.

Click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.14.4 Configure Authentication Mechanisms - SAML 1.x

Use this page to:

- view, add or delete SAML 1.x authentication mechanisms
- map SAML 1.x authentication mechanisms to local authentication mechanisms



Authentication Mechanisms

An authentication mechanism identifies a method or policy for verifying an entity's claimed identity using submitted credentials. An authentication engine is a module implementing a particular authentication method.

Default Authentication Mechanism: oracle:fed:authentication:x509

SAML 1.x Authentication Mechanism Mappings

SAML 1.x Mechanism Name	Local Mechanism Name
urn:oasis:names:tc:SAML:1.0:am:unspecified	oracle:fed:authentication:unspecified
urn:oasis:names:tc:SAML:1.0:am:unspecified	oracle:fed:authentication:internet-protocol
urn:oasis:names:tc:SAML:1.0:am:unspecified	oracle:fed:authentication:telephony

The authentication mechanisms table displays these columns:

- the name of the SAML 1.x authentication mechanism
- the authentication engine currently associated with a mechanism

To change the displayed data, use the View drop-down to select the desired fields.

To add a new authentication mechanism, click **Add**.

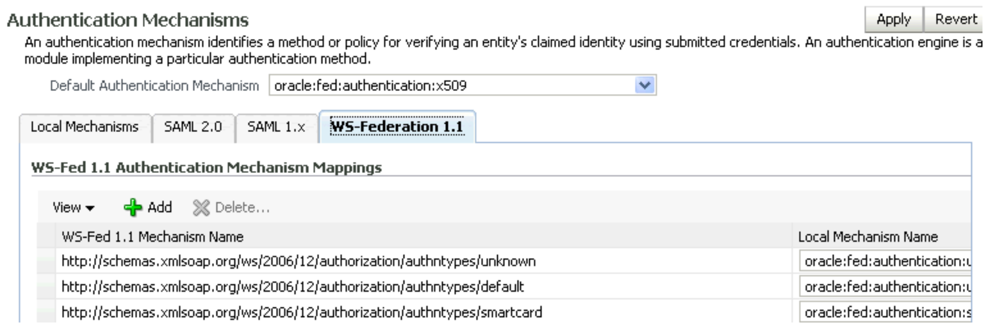
To delete an existing authentication mechanism, select the row for that mechanism and click **Delete**. You will be asked to confirm the deletion.

Click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.14.5 Configure Authentication Mechanisms - WS-Federation 1.1

Use this page to:

- view, add or delete WS-Federation 1.1 authentication mechanisms
- map WS-Federation 1.1 authentication mechanisms to local authentication mechanisms


Authentication Mechanisms

An authentication mechanism identifies a method or policy for verifying an entity's claimed identity using submitted credentials. An authentication engine is a module implementing a particular authentication method.

Default Authentication Mechanism: oracle:fed:authentication:x509

WS-Fed 1.1 Authentication Mechanism Mappings

WS-Fed 1.1 Mechanism Name	Local Mechanism Name
http://schemas.xmlsoap.org/ws/2006/12/authorization/authntypes/unknown	oracle:fed:authentication:u
http://schemas.xmlsoap.org/ws/2006/12/authorization/authntypes/default	oracle:fed:authentication:u
http://schemas.xmlsoap.org/ws/2006/12/authorization/authntypes/smartcard	oracle:fed:authentication:s

The authentication mechanisms table displays these columns:

- the name of the WS-Federation 1.1 authentication mechanism
- the authentication engine currently associated with a mechanism

To change the displayed data, use the View drop-down to select the desired fields.

To add a new authentication mechanism, click **Add**.

To delete an existing authentication mechanism, select the row for that mechanism and click **Delete**. You will be asked to confirm the deletion.

Click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.15 Configuring Authentication Engines

Use this page to configure:

- HTTP headers that are to be used as attributes for a user session

  To configure these headers, click the **Configure** button next to HTTP Header Attributes. A dialog box appears where you can add headers and delete existing headers.

- authentication engines for the Oracle Identity Federation server

This page consists of tabs devoted to individual authentication engines. Updates on any tab are saved as you move to other tabs. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

- Authentication Engines - HTTP Header
- Authentication Engines - Oracle Single Sign-On
- Authentication Engines - Oracle Access Manager 11g
- Authentication Engines - Oracle Access Manager 10g
- Authentication Engines - LDAP Directory
- Authentication Engines - Database Security
- Authentication Engines - Database Table
- Authentication Engines - Infocard
- Authentication Engines - Federated SSO Proxy
- Authentication Engines - JAAS
- Authentication Engines - Custom

5.15.1 Authentication Engines - HTTP Header

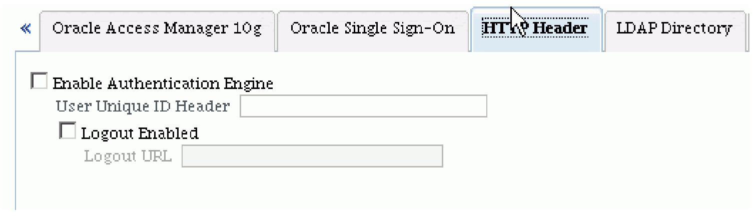
The HTTP Header authentication engine authenticates a user based on the value of an HTTP header.

The typical deployment for such an engine consists of:

- Oracle Identity Federation server deployed in the domain
- a web server (such as Oracle HTTP Server) fronting the WebLogic managed server where Oracle Identity Federation is running (see Section 3.2.1, "Deploying Oracle Identity Federation with Oracle HTTP Server" for details on how to deploy and integrate Oracle HTTP Server if it is not yet installed.)
- a web agent integrated on the web server, protecting the HTTP header authentication engine URL (`http(s)://`*oif-host:oif-port*`/fed/user/authnhttp`)
- a web agent policy for the HTTP header authentication engine URL that instructs the agent to set the user's identity as an HTTP header variable
- Oracle Identity Federation configured to retrieve the HTTP header variable from the HTTP request that contains the user's identity

Since the Web agent protects the HTTP header authentication engine URL, any requests processed by the Oracle Identity Federation server on this URL means that the user was authenticated by the Web Access Management system to which the Web agent belongs.

5.15.1.1 Configuring the HTTP Header Authentication Engine



☐ Enable Authentication Engine
User Unique ID Header
☐ Logout Enabled
Logout URL

The HTTP Header tab contains these fields:

- Enable Authentication Engine - Check this box to enable the engine, and uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with Default Authentication Engine).
- User Unique ID Header - When Oracle Identity Federation uses the HTTP header engine as an authentication engine, a Web agent is integrated with Oracle HTTP Server/Oracle Identity Federation and protects an Oracle Identity Federation URL. The policy domain for the Oracle Identity Federation URL is configured to provide the user identifier as an HTTP header.

  Use this field to specify the name of the HTTP header containing the user identifier provided by the Web agent.
- Logout Enabled - Check this box to enable logouts with this engine. When enabling logouts, related fields include:
  - Logout URL - The is the URL where Oracle Identity Federation needs to redirect the user for the Web Access Management system logout.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

Note: When the user must be redirected to a Web access management URL for logout (to perform some logout operations), you need to configure Oracle Identity Federation by checking Logout Enabled, and entering the URL to which the user is redirected. When Oracle Identity Federation redirects the user to that URL, it appends a return URL as a query parameter; this is the Oracle Identity Federation URL to which the user should be redirected after performing the Web access management logout operations.

Oracle Identity Federation appends the query parameter to the Logout URL referenced by `returnurl`.

5.15.1.2 Configuring HTTP Header Attributes

The HTTP Header Attributes module is invoked after a successful local authentication operation that extracts HTTP headers from the user's HTTP request and saves them as Oracle Identity Federation session attributes. These attributes in turn can be used to populate a SAML assertion when Oracle Identity Federation acts as an identity provider.

Take these steps to configure the HTTP Header Attributes module:

1. In the Authentication Engines section, click **Configure** to manage the HTTP Header Attributes module.

   A window appears that lists the HTTP header attributes that are collected by Oracle Identity Federation after a successful local authentication operation.

2. To add an HTTP header to be collected and saved as a session attribute:

   - Click **Add**.
   - To retrieve an HTTP header, enter its name in the new field. For example, to retrieve the Accept Language header, enter Accept-Language.
   - To retrieve the value of a cookie presented in the HTTP request, enter the name of the cookie, prefixed with `orafed-cookie-`. For example, to retrieve the value of the cookie called userLanguageCookie, enter `orafed-cookie-userLanguageCookie`; at runtime Oracle Identity Federation retrieves the cookie, extracts the value and sets the `orafed-cookie-userLanguageCookie` session attribute.

3. To delete an HTTP header:

   - Select the HTTP header to remove.
   - Click **Delete**.

4. Click **OK** to save the changes.

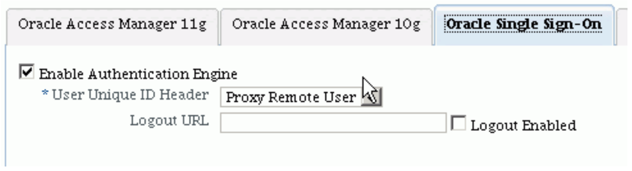
Once the HTTP Header Attributes module is configured, you can configure Oracle Identity Federation to use these session attributes when building assertions, to populate:

- Name Identifier values in the assertion's Subject
- Attributes in the assertion. For example, the "Accept-Language" and "orafed-cookie-userLanguageCookie" can be sent as assertion attributes by listing them in the Attribute Mappings and Filters section of a trusted service provider.

The run-time flow looks like this:

1. Oracle Identity Federation determines that the user needs to be authenticated, and invokes an authentication engine to challenge and identify the user.
2. The user provides some credentials or information by submitting data to the authentication engine URL (for example, username and password POST to the LDAP authentication engine credential processing URL).
3. After the user data is validated, the engine internally forwards the user back to Oracle Identity Federation.
4. The HTTP Header Attributes module analyzes the HTTP request submitted to the authentication engine (the module has access to it because of the internal forward operation). It extracts the required HTTP headers and/or cookie values from this request.
5. Oracle Identity Federation saves the extracted data as session attributes.
6. If configured to do so, during assertion creation Oracle Identity Federation/IdP uses the session attributes to populate the NameID or attribute elements.

5.15.2 Authentication Engines - Oracle Single Sign-On



☑ Enable Authentication Engine
* User Unique ID Header: Proxy Remote User
Logout URL: ☐ Logout Enabled

The tab contains these fields:

- Default Authentication Engine - This is the engine used for authentications. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable Authentication Engine - Check this box to enable the engine, and uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with Default Authentication Engine).
- User Unique ID Attribute - This is the attribute Oracle Identity Federation uses to identify the user.

> **Notes:**
>
> - For every user, this attribute value must equal the attribute value specified for Unique ID Attribute in the User Data Store configuration. For example, if the attribute configured here is `mail`, and the attribute configured as Unique ID Attribute in the user data store configuration is `EmailAddress`, then the value of `mail` in the authentication engine back-end must equal the value of `EmailAddress` in the user data store.
> - The attribute value you configure here must be unique across all users.

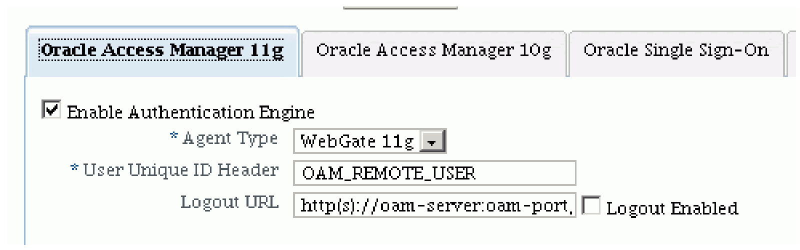
- Logout URL - This is the Oracle Single Sign-On server URL to present at logout.
- Logout Enabled - Check this box to indicate that Oracle Identity Federation will redirect the user to the Oracle SSO Logout URL when the Oracle Identity Federation logout flow is performed.

The logout URL needs to be the Oracle SSO Logout URL:

```
http(s)://sso-host:sso-port/sso/logout
```

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.15.3 Authentication Engines - Oracle Access Manager 11g



☑ Enable Authentication Engine
* Agent Type: WebGate 11g
* User Unique ID Header: OAM_REMOTE_USER
Logout URL: http(s)://oam-server:oam-port, ☐ Logout Enabled

Use this tab to configure the Oracle Access Manager 11*g* authentication engine.

The tab contains these fields:

- Default Authentication Engine - This is the engine used for authentications. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable Authentication Engine - Check this box to enable the engine, and uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with Default Authentication Engine).
- Agent Type - The agent must be one of: Webgate11g, Webgate10g or mod_osso.
- User Unique ID Header - This is the attribute Oracle Identity Federation uses to identify the user. For a mod_osso agent, this is a drop-down list, and is set to `Proxy Remote User` by default. For Webgate10g or Webgate11g, the default is `OAM_REMOTE_USER`.

Notes:

- For every user, this attribute value must equal the attribute value specified for Unique ID Attribute in the user data store. For example, if the attribute configured here is `mail`, and the attribute configured as Unique ID Attribute in the user data store is `EmailAddress`, then the value of `mail` in the authentication engine back-end must equal the value of `EmailAddress` in the user data store.
- The attribute value you configure here must be unique across all users.

- Logout URL - This is the Oracle Access Manager server URL to present at logout. For example:

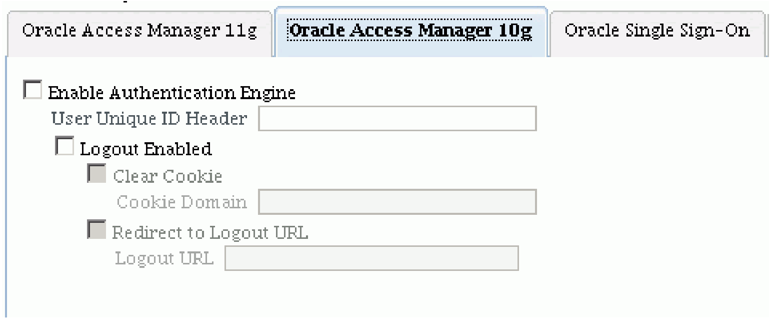
```
http://oam_host:oam_port/oam/server/logout
```

Note: If SP integration with OAM11g engine is enabled and configured for logout, you do not need to configure logout in the OAM11g authentication engine.

- Logout Enabled - Check this box to indicate that Oracle Identity Federation will redirect the user to the Oracle Access Manager Logout URL when the Oracle Identity Federation logout flow is performed.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.15.4 Authentication Engines - Oracle Access Manager 10g



The tab contains these fields:

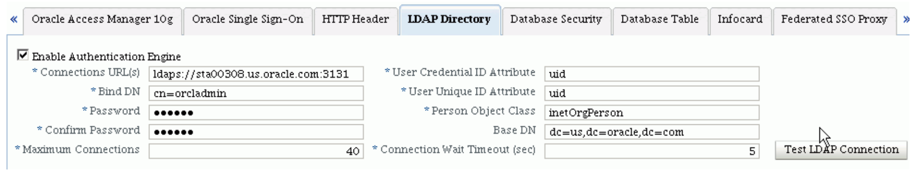
- Default Authentication Engine - This is the engine used for authentications. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable Authentication Engine - Check this box to enable the engine, and uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with Default Authentication Engine).
- User Unique ID Header - When Oracle Identity Federation uses Oracle Access Manager as an authentication engine, WebGate is integrated with Oracle HTTP Server/Oracle Identity Federation and protects an Oracle Identity Federation URL. The policy domain for the Oracle Identity Federation URL is configured so that it will provide the user identifier as an HTTP header.

  Use this field to specify the name of the HTTP header containing the user identifier provided by WebGate.
- Logout Enabled - Check this box to enable logouts with this engine. When enabling logouts, related fields include:
  - Clear Cookie - If checked, resetting the Oracle Access Manager cookie is sufficient for Oracle Identity Federation to log the user out of the Oracle Access Manager domain.
  - Cookie Domain - Cookie domain that Oracle Identity Federation will set when creating the Oracle Access Manager cookie.
  - Redirect to Logout URL - Check this box and fill in the URL if Oracle Identity Federation needs to redirect the user to a specific URL for Oracle Access Manager logout.

– Logout URL - This is the URL to present at logout.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.15.5 Authentication Engines - LDAP Directory



[✓] Enable Authentication Engine

* Connections URL(s)	ldaps://sta00308.us.oracle.com:3131	* User Credential ID Attribute	uid
* Bind DN	cn=orcladmin	* User Unique ID Attribute	uid
* Password	••••••	* Person Object Class	inetOrgPerson
* Confirm Password	••••••	Base DN	dc=us,dc=oracle,dc=com
* Maximum Connections	40	* Connection Wait Timeout (sec)	5

The tab contains these fields:

- Default Authentication Engine - This is the engine used for authentications. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable Authentication Engine - Check this box to enable the engine, and uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with Default Authentication Engine).
- Connection URL(s) - space-delimited list of LDAP server URLs - hostname and port
- Bind DN - This is the DN used by the Oracle Identity Federation server to connect to the LDAP server. For example:

  `cn=fedid,dc=mycompany,dc=com`
- Password - Server password
- Confirm Password - Server password
- Maximum Connections - This is the maximum number of concurrent connections made by Oracle Identity Federation to the LDAP server.
- User Credential ID Attribute - This is the attribute with which Oracle Identity Federation will authenticate the user.

  For example, if the attribute configured here is `mail`, and the value of this attribute for a user is `alice@mycorp.com`, that user will need to authenticate with username `alice@mycorp.com`.

  Note: The attribute value configured here must be unique across all users.

- User Unique ID Attribute - This is the attribute with which Oracle Identity Federation will identify the user.

Notes:

- For every user, the value of this attribute must equal the value of the attribute specified as Unique ID Attribute in the user data store. For example, if the attribute configured here is `mail`, and the attribute configured as Unique ID Attribute in the user data store is `EmailAddress`, then the value of `mail` in the authentication engine back-end must equal the value of `EmailAddress` in the user data store.
- The attribute value configured here must be unique across all users.

- Person Object Class - the LDAP object class representing a user in the LDAP server. Here are examples of the Person Object Class for different types of directory servers:
  - Oracle Internet Directory: inetOrgPerson
  - Sun Java System Directory Server: inetOrgPerson
  - Microsoft Active Directory: user
- Base DN - the node under which LDAP user search will be performed. For example:

  `dc=us,dc=oracle,dc=com`
- Connection Wait Timeout (sec) - the maximum number in seconds to wait until a connection is available, when the maximum number of connections opened by Oracle Identity Federation to the LDAP server has been reached.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

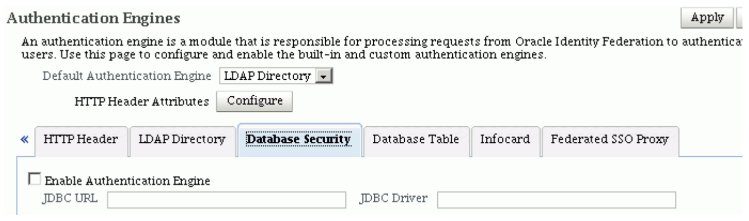
For additional information relevant to configuring LDAP authentication engines, see:

- Section 6.4.1, "Configuring High Availability LDAP Servers"
- Section 5.13.1.4, "Configuring a Redundancy User Data Store" which explains how to configure a load balancer in front of LDAP servers.

5.15.5.1 Configuring Oracle Virtual Directory as the Authentication Engine

Oracle Identity Federation can be integrated with Oracle Virtual Directory; when using Oracle Virtual Directory as the LDAP authentication engine, ensure that the base DN, person object class, unique user id and user description attribute settings are valid for all directory structures connected to Oracle Virtual Directory.

5.15.6 Authentication Engines - Database Security


Authentication Engines

An authentication engine is a module that is responsible for processing requests from Oracle Identity Federation to authentica users. Use this page to configure and enable the built-in and custom authentication engines.

Default Authentication Engine: LDAP Directory

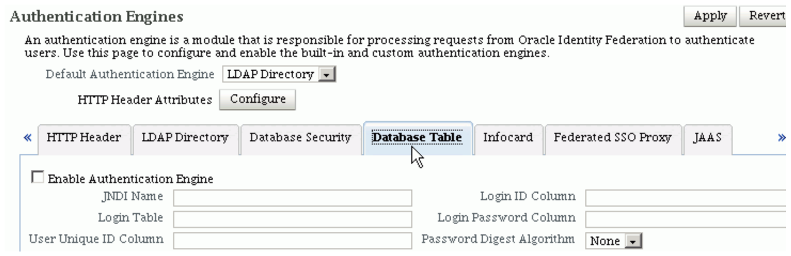
HTTP Header Attributes: Configure

The tab contains these fields:

- Default Authentication Engine - This is the engine used for authentications. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable Authentication Engine - Check this box to enable the engine, and uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with Default Authentication Engine).
- JDBC URL - the connection URL of the database.
- JDBC Driver - Enter the JDBC driver string.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.15.7 Authentication Engines - Database Table



Authentication Engines

An authentication engine is a module that is responsible for processing requests from Oracle Identity Federation to authenticate users. Use this page to configure and enable the built-in and custom authentication engines.

Default Authentication Engine: LDAP Directory

HTTP Header Attributes: Configure

☐ Enable Authentication Engine

JNDI Name		Login ID Column	
Login Table		Login Password Column	
User Unique ID Column		Password Digest Algorithm	None

The tab contains these fields:

- Default Authentication Engine - This is the engine used for authentications. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable Authentication Engine - Check this box to enable the engine, and uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with default authentication engine).
- JNDI Name - The JNDI of the data source created in the Oracle WebLogic Server Administration Console.
- Login Table - The name of the login table.
- Login ID Column - The name of the Login ID column in the Login Table.
- User Unique ID Column - The name of the User ID column in the Login Table.
- Login Password Column - The name of the Login Password column in the Login Table.
- Password Digests Algorithm - The digest algorithm applied to passwords in the Login Table. Select None if the password is stored in clear-text in the database, or select MD5 or SHA1 if the value in the database is an MD5 or SHA1 hash of the password.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.15.7.1 Configuring Oracle Identity Federation for RDBMS Authentication Engine

In order for Oracle Identity Federation to use a database as the authentication engine, this database must have a table, referred to as the login table, that contains user login information. Each user in the login table must have these attributes:

1. Login ID: The unique username with which the user will log in.
2. Login Password: The password with which the user will log in. The value in this column can be the clear-text password or an MD5 or SHA1 hash of the password.
3. User ID: The unique identifier with which the user will be identified in Oracle Identity Federation. The value of the User ID must match the value of the User ID in the user data store.

The attributes Login ID and User ID can be stored in two separate columns, or in one column if the Login ID is to be used as the User ID.

Example 1

Consider the following login table, named "UserLoginInfo1". In this table, the Login ID is under the column "Email", and the User ID is under the column "UserID".

UserID	FirstName	LastName	Password	Email
alice	Alice	Smith	gu*L3nb	alice@mycorp.com
bob	Robert	Jones	aqweb80	bob@mycorp.com
charlie	Charles	Johnson	b23thag	charlie@mycorp.com
david	David	Jones	094bshyq=	david@mycorp.com
robert	Robert	Williams	#)haba*(+	williams@mycorp.com

Example 2

Consider the following login table, named "UserLoginInfo2". In this table, both the Login ID and the User ID are under the "Username" column.

Username	Password	FullName
alice	gu*L3nb	Alice Smith
bob	aqweb80	Robert Jones
charlie	b23thag	Charles Johnson
david	094bshyq=	David Jones
robert	#)haba*(+	Robert Williams

To configure Oracle Identity Federation to use the RDBMS authentication engine, you will need to:

1. Create a JDBC data source.
2. Modify Oracle Identity Federation authentication engine configuration.

Create a JDBC Data Source

Follow these steps to configure an RDBMS user data store:

1. Log in to the WebLogic Administration Console.
2. Navigate to **Services**, then **JDBC**, then **Data Sources**.
3. Click **New**.
4. Choose a Name and a JNDI Name for the new data source, and enter the database information. Choose the WebLogic managed server where Oracle Identity Federation is deployed as the target of this data source.

   See Also: Getting Started with Oracle WebLogic Server Administration Console in the *Oracle Fusion Middleware Administrator's Guide*.

Modify Oracle Identity Federation Authentication Engine Configuration

Follow these steps to configure the RDBMS authentication engine.

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Authentication Engines**.
3. In the **Database Table** tab, select **Enable Authentication Engine** and add the following properties:
   - JNDI Name: The JNDI of the data source created in the WebLogic Administration Console.
   - Login Table: The name of the login table.
   - Login ID Column: The name of the Login ID column in the login table.
   - User Unique ID Attribute: The name of the User ID column in the login table.
   - Login Password Column: The name of the Login Password column in the login table.
   - Password Digest Algorithm: The digest algorithm applied to passwords in the login table.

     Select `None` if the password is stored in clear-text in the database, or select `MD5` or `SHA1` if the value in the database is an MD5 or SHA1 hash of the password.
4. Click **Apply**.

Example 1 Configuration

The configuration corresponding to the table "UserLoginInfo1" in Example 1 above is as follows. Suppose the JNDI name of the data source created for the database is "MyCorpUserDS".

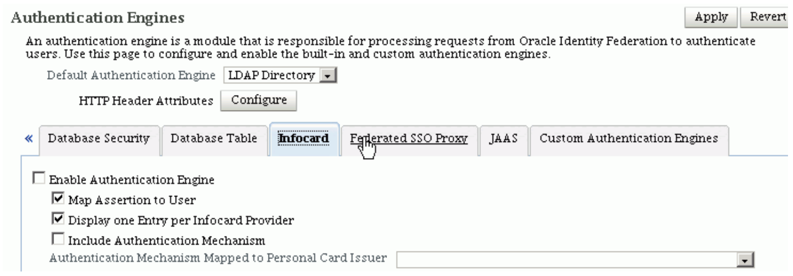
- **JNDI Name:** MyCorpUserDS
- **Login Table:** UserLoginInfo1
- **Login ID Column:** Email
- **User Unique ID Attribute:** UserID
- **Login Password Column:** Password
- **Password Digest Algorithm:** none/MD5/SHA1

Example 2 Configuration

The configuration corresponding to the table "UserLoginInfo2" in Example 2 above is as follows. Suppose the JNDI name of the data source created for the database is "MyCorpUserDS".

- **JNDI Name:** MyCorpUserDS
- **Login Table:** UserLoginInfo2
- **Login ID Column:** Username
- **User Unique ID Attribute:** Username
- **Login Password Column:** Password
- **Password Digest Algorithm:** none/MD5/SHA1

5.15.8 Authentication Engines - Infocard



Authentication Engines

An authentication engine is a module that is responsible for processing requests from Oracle Identity Federation to authenticate users. Use this page to configure and enable the built-in and custom authentication engines.

Default Authentication Engine: LDAP Directory

HTTP Header Attributes: Configure

☐ Enable Authentication Engine

☑ Map Assertion to User

☑ Display one Entry per Infocard Provider

☐ Include Authentication Mechanism

Authentication Mechanism Mapped to Personal Card Issuer:

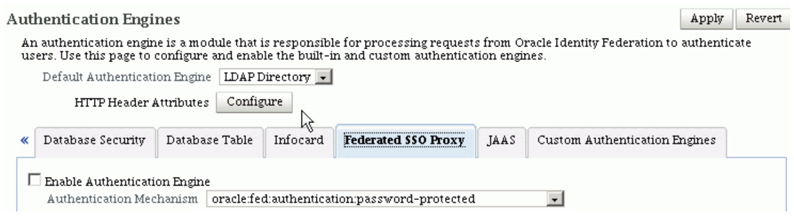
The tab contains these fields:

- Default Authentication Engine - This is the engine used for authentications. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable Authentication Engine - Check this box to enable the engine, and uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with Default Authentication Engine).
- Map Assertion to User - If checked, the incoming assertion is mapped to a user record based on the configuration on the **SAML 2.0 / SAML 1.x Assertion** tab of the **Service Provider** page.
- Display one Entry per Infocard Provider - If checked, Oracle Identity Federation displays an infocard selection option for each Infocard provider configured in the **Federations** page.
- Include Authentication Mechanism - If checked, Oracle Identity Federation adds the authentication mechanism as a required claim, enabling it to request a specific authentication method from the Infocard providers.
- Authentication Mechanism Mapped to Personal Card Issuer - When the Infocard authentication engine is invoked for authentication with the authentication mechanism configured here, the personal issuer card is displayed on the login page. If invoked with an authentication mechanism different from the one configured here, all the Infocard providers are displayed.

See Also: Section 6.12, "Setting up Infocard".

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.15.9 Authentication Engines - Federated SSO Proxy


Authentication Engines

An authentication engine is a module that is responsible for processing requests from Oracle Identity Federation to authenticate users. Use this page to configure and enable the built-in and custom authentication engines.

Default Authentication Engine: LDAP Directory

HTTP Header Attributes: Configure

Enable Authentication Engine

Authentication Mechanism: oracle:fed:authentication:password-protected

The tab contains these fields:

- Default Authentication Engine - This is the engine used for authentications. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable Authentication Engine - Check this box to enable the engine, and uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with Default Authentication Engine).
- Authentication Mechanism - This is the authentication mechanism that Oracle Identity Federation will use to authenticate the user locally when using the Federated SSO proxy.

WARNING: The authentication mechanism specified here *must not* map to the Federated SSO Proxy authentication engine.

See Also: Section 5.15.9.1, "About the Federated SSO Proxy Authentication Engine."

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

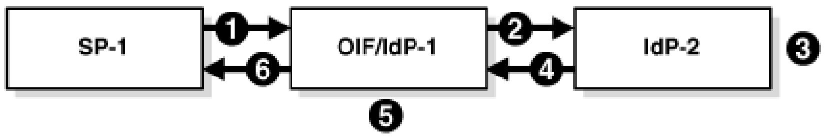
Additional topics include:

- About the Federated SSO Proxy Authentication Engine
- Selecting the Identity Provider to Use
- Configuring the Federated SSO Proxy Authentication Engine

5.15.9.1 About the Federated SSO Proxy Authentication Engine

When an identity provider uses the Federated SSO Proxy authentication engine to authenticate a user, it does this by taking the role of service provider, and initiating a Single Sign-On flow with a second identity provider that authenticates the user.

The flow is as follows:



1. A service provider, SP-1, sends an authentication request to an Oracle Identity Federation identity provider, IdP-1.
2. Oracle Identity Federation/IdP-1 is using the Federated SSO Proxy authentication engine; it selects a trusted identity provider, IdP-2, takes the role of a service provider, and sends a new authentication request for the specified user to IdP-2.
3. IdP-2 authenticates the user.
4. IdP-2 sends back an assertion to Oracle Identity Federation/IdP-1, who will then process this assertion.
5. If necessary, Oracle Identity Federation/IdP-1 authenticates the user locally (for example when a federation creation operation needs to be performed).
6. Oracle Identity Federation/IdP-1 sends back a new assertion to SP-1.

5.15.9.2 Selecting the Identity Provider to Use

When an identity provider using the Federated SSO Proxy authentication engine receives an authentication request from a service provider, it will select a trusted identity provider to which to send a new request. To select the identity provider, Oracle Identity Federation maps the authentication mechanism requested by the service provider (or the default mechanism if the SP did not request one) to an identity provider, and sends a new request to this IdP.

If the mechanism does not map to an identity provider, Oracle Identity Federation uses the default identity provider in configuration. (Refer to Section 5.14.1, "About Authentication Mechanisms" for more information on authentication mechanisms and how protocol-specific methods are mapped to local authentication mechanisms).

For example, suppose that the following mappings from local authentication mechanisms to identity providers are configured:

```
oracle:fed:authentication:internet-protocol -> http://corp-1.com/idp
oracle:fed:authentication:password-protected -> http://corp-2.com/idp
```

and that the default identity provider is: `http://corp-3.com/idp`.

Then, if the service provider requests an authentication method that maps to the `oracle:fed:authentication:internet-protocol`, Oracle Identity Federation selects `http://corp-1.com/idp` as the identity provider, but if the service provider requests `oracle:fed:authentication:password-protected`, Oracle Identity Federation chooses `http://corp-2.com/idp`. If the service provider does not request an authentication method, then Oracle Identity Federation sends the new authentication request to `http://corp-3.com/idp`.

You can define the mappings from local authentication mechanisms to identity providers by following these steps:

See Also: Section 5.5, "Configuring Service Providers"

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Service Provider**.
3. In **Protocol Settings**, click on **Configure SSO Authentication Mechanism to Identity Provider Mapping**.
4. Click **Add**, and select the authentication mechanism and the identity provider to which it maps.

5. When you are done adding mappings, click **OK**. Then click **Apply**.

You can configure the default identity provider by following these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Service Provider**.
3. Select the **Default SSO Identity Provider** and click **Apply**.

5.15.9.3 Configuring the Federated SSO Proxy Authentication Engine

To correctly use the federated SSO proxy authentication engine, you need to configure authentication mechanisms. This might include:

1. Setting the default authentication mechanism
2. Mapping protocol-specific methods to local mechanisms and local mechanisms to authentication engines
3. Mapping local authentication mechanisms to identity providers

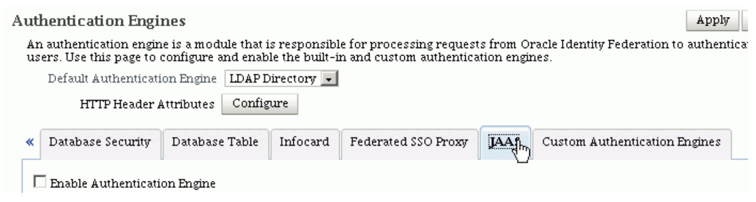
In addition to configuring authentication mechanisms, you will need to configure the federated SSO proxy authentication engine itself. To do this, follow these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Authentication Engines**.
3. In the **Federated SSO Proxy** tab, select **Enable Authentication Engine** and choose the authentication mechanism that will be used to authenticate the user locally when needed.

WARNING: The local authentication mechanism to use when the user needs to be locally authenticated must not be mapped to the Federated SSO Proxy authentication engine. This will create a loop where IdP-1 continuously sends a request to IdP-2. (IdP-1 sends a request to IdP-2 and receives an assertion. It needs to authenticate the user locally, and thus maps the mechanism to the Federated SSO Proxy authentication engine, which will prompt it to send a new request to IdP-2).

Refer to Section 5.14.1, "About Authentication Mechanisms" for more information on authentication mechanisms and how authentication mechanisms are mapped to authentication engines.

5.15.10 Authentication Engines - JAAS


Authentication Engines

An authentication engine is a module that is responsible for processing requests from Oracle Identity Federation to authentica
users. Use this page to configure and enable the built-in and custom authentication engines.

Default Authentication Engine: LDAP Directory

HTTP Header Attributes: Configure

☐ Enable Authentication Engine

The JAAS authentication engine is the default authentication engine for the Oracle Identity Federation server.

Use the **Enable Authentication Engine** check-box to enable or disable this engine.

Since JAAS is the default engine, this box is checked by default. To disable the JAAS authentication engine, another engine must be available to serve as the default engine. If necessary, first set up a different authentication engine, then return to this tab to disable the JAAS engine.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

Note: The JAAS authentication engine does not support logout. This means that after you configure a provider to use the engine, perform single sign-on between IdP and SP, and issue the Oracle Identity Federation logout URL `http://host:port/fed/user/logout`, the user is not logged out and can repeat the SSO flow without having to log in again.

Creating and Adding Users to the oifusers Group

For a user to be authenticated by the JAAS authentication engine, a corresponding user entry must exist in the security realm of the WLS Domain where Oracle Identity Federation is deployed, and must be part of the `oifusers` group.

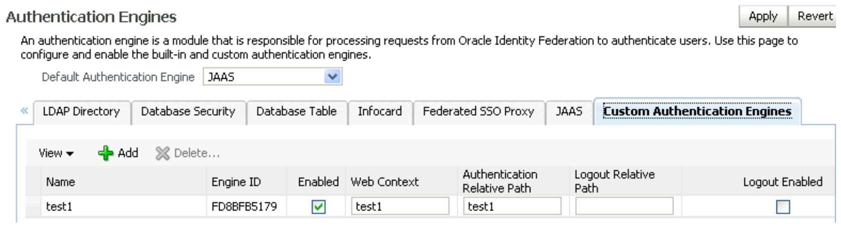
Follow these steps to create the `oifusers` group and add new users.

See Also: Getting Started Managing Oracle Fusion Middleware

1. Log in to Oracle WebLogic Server's Administration Server console.
2. On the left-hand pane, select **Security Realms** and navigate to **myrealm**, then **Users and Groups**, then **Groups**.
3. Click **New** and enter name `oifusers`.
4. Navigate to **Users and Groups**, then **Users**.
5. Click **New** and select a name and password.
6. Click the user you just created and select the **Groups** tab.
7. Select group `oifusers` and move it to the Chosen column. Click **Save**.

To enter additional users, repeat steps 4-7. After the group and users have been created, you must restart the Administration server and managed server where Oracle Identity Federation is running in order for the changes to take effect.

5.15.11 Authentication Engines - Custom


Authentication Engines

An authentication engine is a module that is responsible for processing requests from Oracle Identity Federation to authenticate users. Use this page to configure and enable the built-in and custom authentication engines.

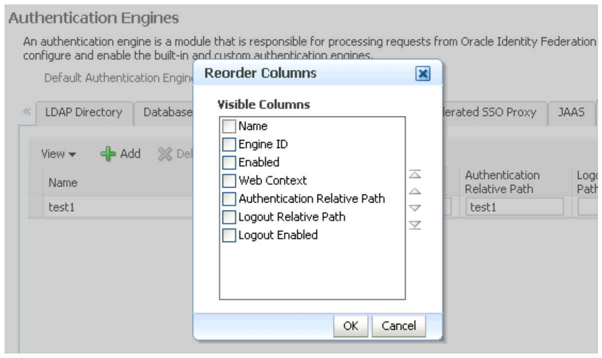
Default Authentication Engine: JAAS

Name	Engine ID	Enabled	Web Context	Authentication Relative Path	Logout Relative Path	Logout Enabled
test1	FD8BFB5179	✓	test1	test1		☐

On this tab, you can set up a custom authentication engine.

View Custom Engines

Use the **View** button to organize the table of custom engines. You can change the column order of the display and specify which fields to include or exclude. The Reorder Columns dialog allows you to select any field and use the arrows to reposition it in the table.


Authentication Engines

An authentication engine is a module that is responsible for processing requests from Oracle Identity Federation
configure and enable the built-in and custom authentication engines.

Add an Engine

Click **Add** to add a new custom engine. You are asked to provide a unique engine name; an Engine ID is automatically generated. Once the engine is added, you can add this information:

- Enabled - Check the box to enable the engine, or uncheck to disable it.
- Web Context - Specifies the Web application context in which your custom authentication engine is deployed.
- Authentication Relative Path - Specifies the path to your custom authentication engine, relative to the Web context.
- Logout Relative Path - Specifies the path to the logout service (if any) for your custom authentication engine, relative to the Web context. For example, "`/auth_engines/myAuthLogout.jsp`".

See Also: Section 10.3, "Creating a Custom Authentication Engine"

The tab contains these fields:

- Default Authentication Engine - This is the engine used for authentications. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable Authentication Engine - Check this box to enable the engine, or uncheck the box to disable the engine. If enabled, this engine appears on the list of available engines (in the list-box associated with Default Authentication Engine).

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.16 Configuring SP Integration Modules

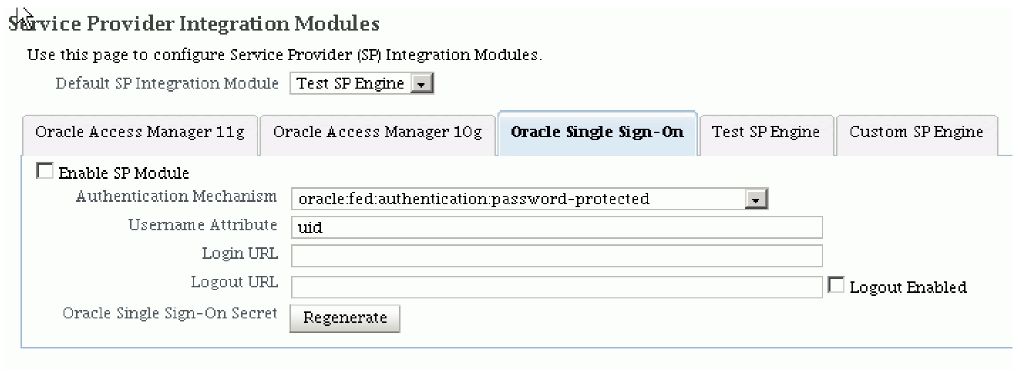
Use this page to configure the SP integration module for Oracle Identity Federation.

This page consists of tabs devoted to individual SP integration module. Updates on any tab are saved as you move to other tabs. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

- SP Integration Module - Oracle Single Sign-On
- SP Integration Module - Oracle Access Manager 11g
- SP Integration Module - Oracle Access Manager 10g
- SP Integration Module - Test SP Engine
- SP Integration Module - Custom

5.16.1 SP Integration Module - Oracle Single Sign-On

Use this tab to configure SP integration for Oracle Single Sign-On.


Service Provider Integration Modules

Use this page to configure Service Provider (SP) Integration Modules.

Default SP Integration Module: Test SP Engine

☐ Enable SP Module

Authentication Mechanism	oracle:fed:authentication:password-protected
Username Attribute	uid
Login URL	
Logout URL	☐ Logout Enabled
Oracle Single Sign-On Secret	Regenerate

The tab contains these fields:

- Default SP Integration module - This is the module used for integration at the service provider. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable SP module - Check this box to enable the module, and uncheck the box to disable the module. If enabled, this module appears on the list of available modules (in the list-box associated with Default SP Integration module).
- Authentication mechanism - authentication mechanism that will be used to locally authenticate users if Federated Identities are used during Federation SSO and if a Federation Record needs to be created during the SSO operation.
- Username Attribute - Username Attribute that Oracle Identity Federation needs to provide to Oracle SSO. Default is `uid`.
- Login URL - This is the Oracle Single Sign-On server URL to present at login. For example:

  ```
  http://sso_host:sso_port/sso/auth
  ```

- Logout URL - This is the Oracle Single Sign-On server URL to present at logout. For example:

  ```
  http://sso_host:sso_port/sso/logout
  ```

- Logout Enabled – Enable/disable logout for the Oracle Single Sign-On application.

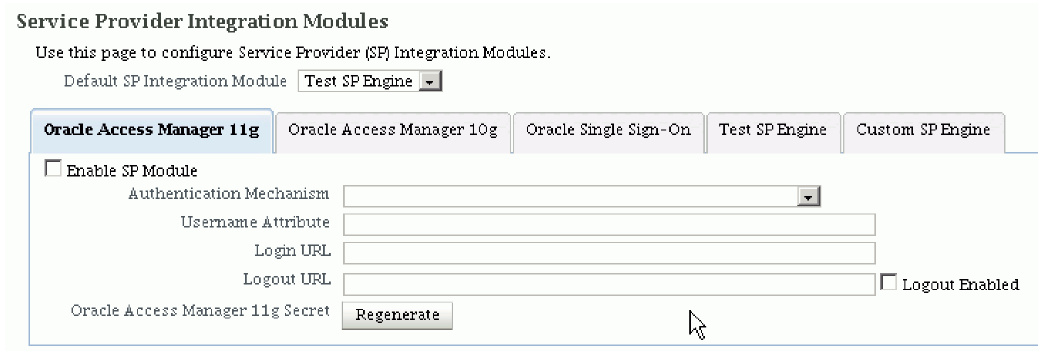
The **Regenerate** button would create an encryption key that will be saved in a file and provided to the Oracle SSO Server.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.16.2 SP Integration Module - Oracle Access Manager 11g

Use this tab to configure SP integration for Oracle Access Manager 11*g*.

Note: In release 11.1.1.7.0, an additional step is needed before you regenerate the OAM 11g Secret Key from this page. See the release notes for details.



Service Provider Integration Modules

Use this page to configure Service Provider (SP) Integration Modules.

Default SP Integration Module: Test SP Engine

The tab contains these fields:

- Default SP Integration module - This is the module used for integration at the service provider. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable SP module - Check this box to enable the module, and uncheck the box to disable the module. If enabled, this module appears on the list of available modules (in the list-box associated with Default SP Integration module).
- Authentication mechanism - This is the authentication mechanism that will be used to locally authenticate users if federated identities are used during federation SSO and if a federation record needs to be created during the SSO operation.
- Username Attribute - This is the username attribute that Oracle Identity Federation needs to provide to Oracle Access Manager. The default is `cn`.
- Login URL - This is the Oracle Access Manager server URL to present at login. For example:

  ```
  http://oam_host:oam_port/oam/server/dap/cred_submit
  ```

- Logout URL - This is the Oracle Access Manager server URL to present at logout. For example:

  ```
  http://oam_host:oam_port/oam/server/logout
  ```

- Logout Enabled – Enable/disable logout for Oracle Access Manager.

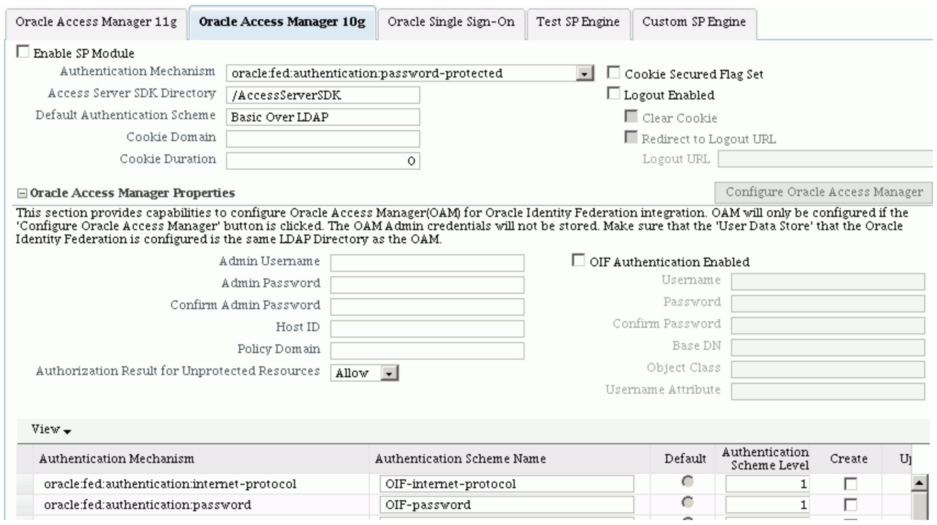
The **Regenerate** button generates a keystore file that contains the keys used to encrypt and decrypt the tokens that are exchanged between the Oracle Access Manager and Oracle Identity Federation servers.

Note: In release 11.1.1.7.0, an additional step is needed before you regenerate the OAM 11g Secret Key from this page. See the release notes for details.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.16.3 SP Integration Module - Oracle Access Manager 10g

Use this tab to configure SP integration for Oracle Access Manager.



Oracle Access Manager 11g | **Oracle Access Manager 10g** | Oracle Single Sign-On | Test SP Engine | Custom SP Engine

☐ Enable SP Module

Authentication Mechanism	oracle:fed:authentication:password-protected
Access Server SDK Directory	/AccessServerSDK
Default Authentication Scheme	Basic Over LDAP
Cookie Domain	
Cookie Duration	0

☐ Cookie Secured Flag Set
☐ Logout Enabled
☐ Clear Cookie
☐ Redirect to Logout URL
Logout URL

Oracle Access Manager Properties — Configure Oracle Access Manager

This section provides capabilities to configure Oracle Access Manager(OAM) for Oracle Identity Federation integration. OAM will only be configured if the 'Configure Oracle Access Manager' button is clicked. The OAM Admin credentials will not be stored. Make sure that the 'User Data Store' that the Oracle Identity Federation is configured is the same LDAP Directory as the OAM.

Admin Username	
Admin Password	
Confirm Admin Password	
Host ID	
Policy Domain	
Authorization Result for Unprotected Resources	Allow

☐ OIF Authentication Enabled

Username	
Password	
Confirm Password	
Base DN	
Object Class	
Username Attribute	

View

Authentication Mechanism	Authentication Scheme Name	Default	Authentication Scheme Level	Create	U
oracle:fed:authentication:internet-protocol	OIF-internet-protocol	○	1	☐	
oracle:fed:authentication:password	OIF-password	○	1	☐	

See Also: Section 3.2.4.3, "Integrate Oracle Access Manager as an SP Integration Module" for details about configuring Oracle Access Manager as an SP integration engine.

The tab contains these fields:

- Default SP Integration module - This is the module used for integration at the service provider. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable SP module - Check this box to enable the module, and uncheck the box to disable the module. If enabled, this module appears on the list of available modules (in the list-box associated with Default SP Integration module).
- Authentication mechanism - The authentication mechanism that will be used to locally authenticate users if federated identities are used during federation SSO and if a federation record needs to be created during the SSO operation.
- Access Server SDK directory – Directory location of Access Server SDK. Absolute path or relative to `DOMAIN_HOME`.
- Default Authentication Scheme – The authentication scheme in Oracle Access Manager that will be used as default scheme for the policy created by Oracle Identity Federation.
- Cookie Domain - Cookie domain that Oracle Identity Federation sets when creating the Oracle Access Manager cookie.
- Cookie duration - For a persistent cookie, the time in minutes during which the cookie will be valid; for a session cookie, enter 0.
- Cookie Secured Flag Set - Check whether the cookie should be marked as secure: in this case, the browser will send the cookie over an HTTPS connection.

- Logout Enabled – Enable/disable logout for the Oracle Access Manager application.
- Clear Cookie – If checked, resetting the Oracle Access Manager cookie is enough for Oracle Identity Federation to log the user out of the Oracle Access Manager domain.
- Redirect to Logout URL - Check Redirect to Logout URL and fill in the URL if Oracle Identity Federation needs to redirect the user to a specific URL for Oracle Access Manager logout.

Note: When the user needs to be redirected to an Oracle Access Manager URL for logout (in case Oracle Access Manager needs to perform extra operations), you need to configure Oracle Identity Federation by checking the Redirect to Logout URL box, and entering the URL to which the user is redirected. When Oracle Identity Federation redirects the user to that URL, it appends a return URL as a query parameter; this is the Oracle Identity Federation URL to which the user is redirected after performing the extra Oracle Access Manager operations.

The query parameter to be appended to the Oracle Access Manager logout URL is referenced by `returnurl`.

The tab contains these fields needed to integrate Oracle Access Manager with Oracle Identity Federation:

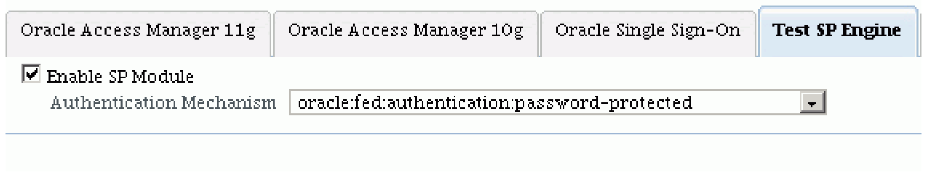
- Oracle Access Manager credentials needed to configure the policy domain
- Oracle Identity Federation account information to enable that server to authenticate itself to Oracle Identity Federation

For details about using these features, see Section 3.2.4.3, "Integrate Oracle Access Manager as an SP Integration Module" and Section 3.2.5, "Oracle Identity Federation/SP Authenticating to Oracle Access Manager".

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.16.4 SP Integration Module - Test SP Engine

Use this tab to configure SP integration for the test SP engine.



☑ Enable SP Module
Authentication Mechanism: oracle:fed:authentication:password-protected

The tab contains these fields:

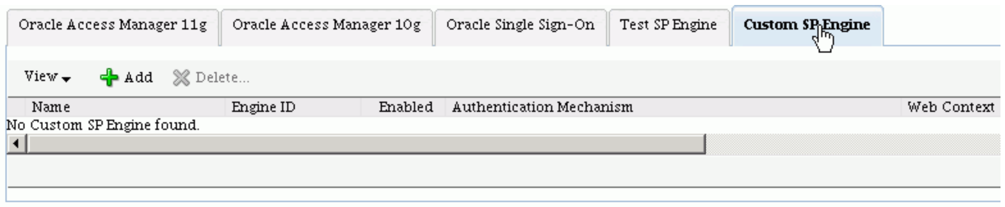
- Default SP Integration module - This is the module used for integration at the service provider. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.

- Enable SP module - Check this box to enable the module, and uncheck the box to disable the module. If enabled, this module appears on the list of available modules (in the list-box associated with Default SP Integration module).
- Authentication mechanism - authentication mechanism that will be used to locally authenticate users if federated identities are used during federation SSO and if a federation record must be created during the SSO operation.

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

5.16.5 SP Integration Module - Custom

Use this tab to configure SP integration for the custom SP engine.



Name	Engine ID	Enabled	Authentication Mechanism	Web Context

No Custom SP Engine found.

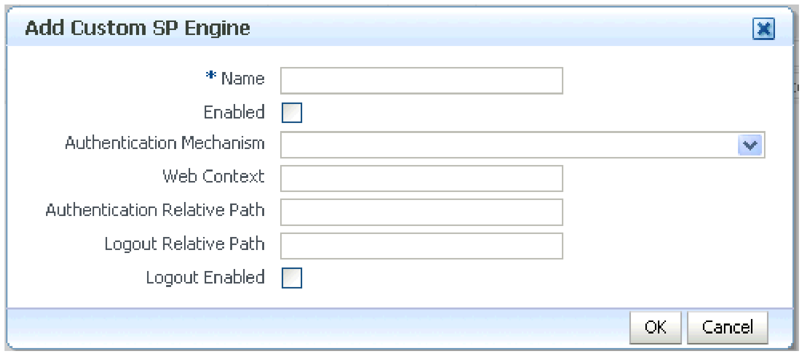
The tab contains these fields:

- Default SP Integration module - This is the module used for integration at the service provider. The list-box contains all the currently enabled engines; selecting an engine from the list makes it the default engine.
- Enable SP module - Check this box to enable the module, and uncheck the box to disable the module. If enabled, this module appears on the list of available modules (in the list-box associated with Default SP Integration module).

View SP integration modules

Use the **View** button to organize the table of SP integration modules. You can change the column order of the display and specify which fields to include or exclude. The Reorder Columns dialog allows you to select any field and use the arrows to reposition it in the table.

Add an Engine



Click the **Add** button to add a new custom engine. You are asked to provide a unique engine name; an Engine ID is automatically generated. Once the engine is added, you can add this information:

- Enabled - Check the box to enable the engine, or uncheck to disable it.

- Authentication mechanism – authentication mechanism to use if a local authentication procedure needs to occur during the assertion processing
- Web Context - Specifies the `contextPath` of the SP integration engine in the **Web Context** field. For example: `/engine`.
- Authentication Relative Path - Specifies the relative path of the login service of the SP integration engine in the **Login Relative Path** field. For example: `/application.jsp`.
- Logout Relative Path -Specifies the relative path of the logout service of the SP integration engine in the **Logout Relative Path** field.

See Also: Section 10.4, "Creating a Custom SP Integration Engine".

Updates you make on this tab are saved if you move to tabs for other authentication engines. When you are done, click **Apply** to save the changes, or **Revert** to reset the data to its previous state.

6

Additional Server Configuration

This chapter contains additional topics pertaining to Oracle Identity Federation server configuration and management. These topics include:

6.1 Setting up Single Sign-On Services

There are several ways to perform a federated single sign-on (SSO) operation, depending on the back-end in use and where the flow is initialized. Table 6–1 shows the possible combinations:

Table 6–1 Federated Single Sign-On Combinations

Combination	Flow
Oracle Single Sign-On	User accesses a resource protected by `mod_osso`, triggering single sign-on, with Oracle Identity Federation acting as SP.
Oracle Access Manager	User accesses a resource protected by webgate, triggering single sign-on, with Oracle Identity Federation acting as SP.
SP-initiated single sign-on	User initiates single sign-on by directly accessing an Oracle Identity Federation URL, with Oracle Identity Federation acting as SP.
IdP-initiated single sign-on	User initiates a single sign-on by directly accessing an Oracle Identity Federation URL, with Oracle Identity Federation acting as IdP.

This section explains how to configure the different combinations:

- Oracle Single Sign-On
- Oracle Access Manager
- SP-initiated SSO
- IdP-initiated SSO

6.1.1 Oracle Single Sign-On

Oracle Single Sign-On can be configured to trigger an SSO operation when requesting a resource protected by `mod_osso`.

To achieve this, the Oracle Single Sign-On partner application must be defined and must be protected by `mod_osso`. The partner application must also be configured to use the SSO Security level associated with the SASSO Authentication plug-in. You do this by editing the `ORACLE_HOME/sso/conf/policy.properties` file of the Oracle Single Sign-On deployment, and setting the partner application (defined by its hostname and port) to the same security level as the `SASSOAuthLevel` property.

For example:

```
MediumHighSecurity_AuthPlugin = oracle.security.sso.server.auth.SASSOAuth
MediumSecurity_AuthPlugin = oracle.security.sso.server.auth.SSOServerAuth
...
www.app.com\:7890 = MediumHighSecurity
...
SASSOAuthnUrl= http\://oif-hostname\:oif-port/fed/user/sposso
SASSOLogoutUrl = http\://oif-hostname\:oif-port/fed/user/spsloosso
SASSOAuthLevel = MediumHighSecurity
```

Save the file and restart the Oracle Single Sign-On server to apply the changes.

The next time a user attempts an unauthenticated access to the protected resource, the user is redirected to Oracle Identity Federation where an SSO operation occurs.

URL Query Parameters

When requesting the protected resource, it is possible to specify URL query parameters that Oracle Identity Federation can use to perform the SSO operation. The parameters are:

- providerid - This is the identifier of the identity provider to use to perform the SSO operation (optional). If missing, the default SSO provider, set in Fusion Middleware Control by navigating to **Service Provider**, then **Common**, then **Default SSO Identity Provider**, is used.
- federationid - This is the identifier of the affiliation to use for the SSO (optional).

An example of such a URL is:

```
http://protected_app:port/path?providerid=http%3A%2F%2Fidp.com
```

Check that the URL query parameter values are correctly URL-encoded.

> **See Also:** Section 6.2, "Working with Affiliations" for more information

Refer to the Oracle Single Sign-On documentation for details about SSO configuration.

6.1.2 Oracle Access Manager

Oracle Access policies can be set up to initiate an SSO operation when the user requests a resource protected by the Oracle Access Manager WebGate agent.

To do this, use the Oracle Access Policy Manager to set up a policy domain or policy that protects the resource. When creating the authentication rule for the policy domain or policy, select the Fed SSO authentication scheme. Oracle Identity Federation automatically creates this authentication scheme when it is configured to use Oracle Access. The scheme initiates the single sign-on operation when the resource is accessed, resulting in a session for the local user associated with the federated user. Set up the authorization rules and expression for the policy domain or policy to allow access for the resulting local user.

URL Query Parameters

When requesting the protected resource, it is possible to specify URL query parameters that Oracle Identity Federation can use to perform the SSO operation. The parameters are:

- providerid - This is the identifier of the identity provider to use to perform the SSO operation (optional). If missing, the default SSO provider, set in Fusion Middleware Control by navigating to **Service Provider** , then **Common**, then **Default SSO identity provider**, is used.
- federationid - This is the identifier of the affiliation to use for the SSO (optional).

> **See Also:** Section 6.2, "Working with Affiliations" for more information

An example of such a URL is:

```
http://protected_app:port/path?providerid=http%3A%2F%2Fidp.com
```

Check that the URL query parameter values are correctly URL-encoded.

Refer to the *Oracle Access Manager Identity and Common Administration Guide* for details about SSO configuration.

6.1.3 SP-initiated SSO

When Oracle Identity Federation server is acting as a service provider, a user can initiate an SSO operation by directly requesting a service at the Oracle Identity Federation/SP instance.

The URL to be requested on Oracle Identity Federation is:

```
http(s)://OIF_host:OIF_port/fed/sp/initiatesso
```

URL Query Parameters

It is possible to specify URL query parameters when requesting the URL:

- providerid - This is the identifier of the identity provider to use to perform the SSO operation (optional). If missing, the default SSO provider, set in Fusion Middleware Control by navigating to **Service Provider**, then **Common**, then **Default SSO Identity Provider**, is used.
- federationid - This is the identifier of the affiliation to use for SSO (optional).

  See Also: Section 6.2, "Working with Affiliations" for more information

- returnurl - This is the URL to which the user is sent after a successful SSO operation. It is required if the Unsolicited Relay State property, set in Fusion Middleware Control by navigating to **Federations**, then **Service Providers (Common)**, is empty.

An example of such a URL is:

```
http://oif_host:oif_
port/fed/sp/initiatesso?providerid=http%3A%2F%2Fidp.com&returnurl=http%3A%2F%FProt
ectedAppHost%2FProtectedAppPath
```

Check that the query parameter values are correctly URL-encoded.

6.1.4 IdP-initiated SSO

Oracle Identity Federation provides the ability to initiate an SSO operation by directly requesting a URL at the Oracle Identity Federation instance acting as an IdP; this is called an SSO IdP-initiated operation.

URL Format

The URL to be requested on Oracle Identity Federation is of the form:

```
http(s)://oif_host:oif_port/fed/idp/initiatesso
```

URL Query Parameters

It is possible to specify query parameters when requesting that URL:

- providerid - This is the identifier of the service provider to use to perform the SSO operation (optional).
- federationid - This is the identifier of the affiliation to use for the SSO (optional).

  See Also: Section 6.2, "Working with Affiliations" for more information

- returnurl - This is the URL to which the user is sent after a successful SSO operation (optional).
- acsurl - This is the Assertion Consumer Service URL. The value should contain the URL encoded value of the Assertion Consumer Service URL.

  OIF/IdP compares the specified URL to the Assertion Consumer Service URLs listed in the SP metadata; it must match one of the ACS URLs in the metadata.

Check that the query parameter values are correctly URL-encoded.

An example of such a URL is:

```
http://oif_host:oif_
port/fed/idp/initiatesso?providerid=http%3A%2F%2Fsp.com&returnurl=http%3A%2F%FProt
ectedAppHost%2FProtectedAppPath
```

6.2 Working with Affiliations

The run-time functioning of affiliations depends on whether the Oracle Identity Federation server is acting as an IdP or an SP.

Oracle Identity Federation Acting as IdP

When Oracle Identity Federation is an IdP, provided the affiliation/SP is present and enabled in the circle of trust, the Oracle Identity Federation server is ready to process any requests originating from service providers using the affiliation.

Oracle Identity Federation Acting as SP

As an SP, you can trigger a single sign-on operation with an IdP using an affiliation to which the SP belongs. To do so, include a `federationid` query parameter in the URL protected by the IdM back-end, and set the parameter value to the affiliation ID.

For example with an Oracle Single Sign-On back-end, assuming that a resource is protected by `mod_osso` and configured for Oracle Identity Federation authentication, requesting the URL of this resource with the `federationid` query parameter instructs Oracle Identity Federation to use an affiliation when performing single sign-on with a peer IdP. Here is an example of such a URL:

```
http://protected_res_host:protected_res_
port/path?federationid=http%3A%2F%Faffiliationid
```

It is also possible to directly access the `http://oif_host:oif_ port/fed/sp/initiatesso` URL with the same `federationid` query parameter. In this case, Oracle Identity Federation triggers a single sign-on operation, and uses the **Unsolicited SSO RelayState** for the peer IdP as the URL to which the user is redirected after successful authentication.

Note: The **Unsolicited SSO RelayState** is set by navigating to **Federations**, then **Edit Trusted Provider** in Fusion Middleware Control.

6.3 Additional LDAP Configuration

This section contains topics for LDAP configuration and maintenance:

- Configuring the LDAP Inactivity Setting
- Configuring the LDAP Read Timeout Setting

- ECID Support for LDAP Connections

6.3.1 Configuring the LDAP Inactivity Setting

When Oracle Identity Federation is integrated with high availability LDAP servers to serve as user data store, federation data store, or authentication engine, the server keeps a pool of LDAP connections that can be re-used for subsequent requests.

Over time, the LDAP server may close some connections due to a long inactivity period, and if left unchecked, this can result in errors and a degradation of performance in Oracle Identity Federation.

You can set an inactivity attribute that tells Oracle Identity Federation how long an LDAP connection should be kept in a pool before being removed due to inactivity. By default the inactivity timeout is set to 300 seconds.

To set the inactivity settings for Oracle Identity Federation, enter the `WLST` script environment for Oracle Identity Federation and set the following properties:

- Set the `ldapconnectioninactivitytimeout long` property from the `authnengines` group to the inactivity timeout in seconds to configure the LDAP Authentication Engine Inactivity Timeout as in this example:

  ```
  setConfigProperty('authnengines', 'ldapconnectioninactivitytimeout', '300',
  'long')
  ```

- Set the `userldapconnectioninactivitytimeout long` property from the `datastore` group to the inactivity timeout in seconds to configure the LDAP user data store Inactivity Timeout as in this example:

  ```
  setConfigProperty('datastore',
  'userldapconnectioninactivitytimeout', '300', 'long')
  ```

- Set the `fedldapconnectioninactivitytimeout long` property from the `datastore` group to the inactivity timeout in seconds to configure the LDAP Federation Data Store Inactivity Timeout as in this example:

  ```
  setConfigProperty('datastore',
  'fedldapconnectioninactivitytimeout', '300', 'long')
  ```

6.3.2 Configuring the LDAP Read Timeout Setting

When Oracle Identity Federation is integrated with LDAP servers for user data store, federation data store or LDAP authentication engine, the server communicates with the LDAP directory to retrieve user attributes, authenticate users, look up users and perform related operations.

Sometimes, the LDAP server can become unresponsive, causing the thread/user to wait for a response or an error. To avoid waiting too long for an error when the server is not responding, Oracle Identity Federation sets a read timeout property on the LDAP connection: if the LDAP server does not respond before the read timeout period, an error is generated, Oracle Identity Federation closes the connection, opens a new one and re-issues the LDAP command.

It is possible to set the read timeout setting to tell the Oracle Identity Federation server how long to wait for data from the LDAP server. By default the read timeout is set to 10 seconds.

To set the read timeout settings for Oracle Identity Federation, enter the `WLST` script environment for Oracle Identity Federation, and set the following properties if necessary (examples are included):

- Set the `ldapconnectionreadtimeout` long property from the `authnengines` group to the read timeout in seconds to configure the LDAP Authentication Engine Read Timeout:

  ```
  setConfigProperty('authnengines', 'ldapconnectionreadtimeout', 'long',
  '10')
  ```

- Set the `userldapconnectionreadtimeout` long property from the `datastore` group to the read timeout in seconds to configure the LDAP user data store read timeout:

  ```
  setConfigProperty('datastore',
  'userldapconnectionreadtimeout', 'long', '10')
  ```

- Set the `fedldapconnectionreadtimeout` long property from the `datastore` group to the read timeout in seconds to configure the LDAP federation data store read timeout:

  ```
  setConfigProperty('datastore', 'fedldapconnectionreadtimeout', 'long', '10')
  ```

6.3.3 ECID Support for LDAP Connections

Oracle Identity Federation 11g supports execution context ID (ECID) for DMS and audit purposes.

When creating an LDAP connection with Oracle Internet Directory, Oracle Identity Federation can pass the ECID context to the OID LDAP Connection.

This feature is disabled by default. To enable (disable) the feature, set the following properties to true (false):

- Set the `ldapuseecid` boolean property in `authnengines` group of `config` for LDAP authn engine
- Set the `userldapuseecid` boolean property in `authnengines` group of config for the LDAP user store
- Set the `fedldapuseecidboolean` property in `authnengines` group of config for LDAP federation data store

Note: The LDAP server (for which ECID support is being enabled) must be Oracle Internet Directory 11*g* Release 1 (11.1.1) or later.

6.4 Additional Configuration for High Availability

This section contains additional topics for high availability configuration:

- Configuring High Availability LDAP Servers
- Configuring the HTTP Session State Sleep/Retry Interval
- Configuring Oracle Identity Federation HA in SSL mode

6.4.1 Configuring High Availability LDAP Servers

By default, Oracle Identity Federation is not configured to integrate with a high availability LDAP server. To integrate Oracle Identity Federation with HA LDAP servers to serve as user data store, federation data store, or authentication engine, Oracle Identity Federation needs to be configured for based on the LDAP server's function.

Enter the `WLST` script environment for Oracle Identity Federation, then set the following properties as needed:

- To integrate the user data store with an HA LDAP server, set the `userldaphaenabled` boolean property from the `datastore` group to `true`; otherwise set it to `false`:

  ```
  setConfigProperty('datastore',
  'userldaphaenabled', 'true', 'boolean')
  ```

- To integrate the Federation Data Store with an HA LDAP server, set the `fedldaphaenabled` boolean property from the `datastore` group to `true`; otherwise set it to `false`:

  ```
  setConfigProperty('datastore', 'fedldaphaenabled',
  'true', 'boolean')
  ```

- To integrate the LDAP authentication engine with an HA LDAP server, set the `ldaphaenabled` boolean property from the `authnengines` group to `true`; otherwise set it to `false`:

  ```
  setConfigProperty('authnengines',
  'ldaphaenabled', 'true', 'boolean')
  ```

6.4.2 Configuring the HTTP Session State Sleep/Retry Interval

When Oracle Identity Federation is deployed in HA mode in a cluster, it can be configured so that the User HTTP session state is replicated across the Oracle WebLogic servers where Oracle Identity Federation is running.

By default the HTTP session state replication is disabled for Oracle Identity Federation. To enable it, refer to Cluster-Wide Configuration Changes in the *Oracle Fusion Middleware High Availability Guide*.

The rest of this section provides some additional configuration for Oracle Identity Federation when HTTP session state replication is enabled.

Note: For performance reasons, disabling HTTP session state replication is the preferred approach.

This additional configuration allows the user to visit different Oracle Identity Federation servers without encountering any errors during processing of a federation request.

Sometimes, the HTTP session state is not replicated fast enough between server instances, generating an error when the user accesses a service in an Oracle Identity Federation instance to which the state has not yet been copied.

You can choose one of two options to avoid this issue:

- Enable sticky sessions on the load balancer to force a specific user to visit the same Oracle WebLogic Server managed server every time it sends an HTTP request;

  or

- Set additional configuration properties in Oracle Identity Federation so that, when the server detects that the HTTP session state has not been replicated yet, it can wait to allow the information to be copied. You can enable this feature and configure the wait time.

> **Note:** For performance reasons, enabling sticky sessions is the preferred approach.

To enable and set the wait time for the User HTTP Session State replication setting for Oracle Identity Federation, enter the `WLST` script environment for Oracle Identity Federation and set the following properties:

- To configure Oracle Identity Federation to wait for the session state to be replicated, set the `sessionreplicationenabled` boolean property from the `serverconfig` group to `true`, otherwise set it to `false`:

```
setConfigProperty('serverconfig',
'sessionreplicationenabled', 'true', 'boolean')
```

- Set the `sessionreplicationtimeout long` property from the `serverconfig` group to the wait time in milliseconds, for example:

```
setConfigProperty('serverconfig', 'sessionreplicationtimeout', '2000', 'long')
```

6.4.3 Configuring Oracle Identity Federation HA in SSL mode

In a high availability environment with two (or more) Oracle Identity Federation servers mirroring one another and a load balancer at the front-end, there are two ways to set up SSL:

- Configure SSL on the load balancer, so that the SSL connection is between the user and the load balancer. In this case, the keystore/certificate used by the load balancer has a CN referencing the address of the load balancer.

  The communication between the load balancer and the WLS/Oracle Identity Federation can be clear or SSL (and in the latter case, Oracle WebLogic Server can use any keystore/certificates, as long as these are trusted by the load balancer).

- Configure SSL on the Oracle Identity Federation servers, so that the SSL connection is between the user and the Oracle Identity Federation server. In this case, the CN of the keystore/certificate from the Oracle WebLogic Server/Oracle Identity Federation installation needs to reference the address of the load balancer, as the user will connect using the hostname of the load balancer, and the Certificate CN needs to match the load balancer's address.

  In short, the keystore/certificate of the SSL endpoint connected to the user (load balancer or Oracle WebLogic Server/Oracle Identity Federation) needs to have its CN set to the hostname of the load balancer, since it is the address that the user will use to connect to Oracle Identity Federation.

6.5 Additional RDBMS Configuration

This section contains additional topics for RDBMS configuration for Oracle Identity Federation:

- Configuring RDBMS Session Cache
- Configuring RDBMS Data Compression

6.5.1 Configuring RDBMS Session Cache

When Oracle Identity Federation is using an RDBMS to store the user session objects, the server uses a caching mechanism to improve performance at runtime: the server keeps a reference to recently used session objects in memory to avoid read access to the database.

Note: This is a critical feature, since a given user's session is accessed multiple times when performing an SSO operation.

You can configure the maximum number of session entries in the cache, and the maximum time the session is present in the cache before it is cleared. By default, Oracle Identity Federation server caches a maximum of 25,000 session entries, for a maximum time of 300 seconds

It is important to set an optimal timeout, especially in cluster mode where the session can be destroyed by another Oracle Identity Federation server if:

- a load balancer is used without sticky sessions
- SOAP Logout is enabled

To set maximum number of entries and the timeout settings for Oracle Identity Federation, enter the `WLST` script environment for Oracle Identity Federation and set the properties as in the following examples:

- Set the `transientrdbmssessioncachesize long` property from the `datastore` group to the maximum entries:

  ```
  setConfigProperty('datastore', 'transientrdbmssessioncachesize', '25000',
  'long')
  ```

- Set the `transientrdbmssessioncachetimeout long` property from the `datastore` group to the cache timeout in seconds:

  ```
  setConfigProperty('datastore', 'transientrdbmssessioncachetimeout', '300',
  'long')
  ```

6.5.2 Configuring RDBMS Data Compression

To decrease the amount of data to be stored in an RDBMS, Oracle Identity Federation provides the capability to compress the data before storing it to the database.

When Oracle Identity Federation is integrated with an RDBMS to store its user session Data or Message Data, the decision on when to compress data can be important.

There are three kinds of data that can be compressed:

Note: Liberty 1.x support is deprecated.

- AuthnRequest for SSO Artifact profile: when Oracle Identity Federation acts as an IdP for Liberty 1.x protocol, the server stores the AuthnRequest message in the RDBMS when the artifact profile is used. If Liberty 1.x is not used, this data should not be compressed. By default compression is disabled.
- Assertion Response for SSO Artifact profile: when Oracle Identity Federation acts as an IdP for SSO protocols, the server stores the Response message containing the

assertion in the RDBMS when the artifact profile is used. This should be enabled if attributes are contained in the assertion. By default compression is enabled.

- User Session Data: Oracle Identity Federation stores some session data related to the user at runtime. If several attributes are stored in the user session (set by a custom Authentication Engine, or because the Attributes assertion storage was enabled when Oracle Identity Federation is an SP), then compression should be used. By default compression is disabled.

To configure Oracle Identity Federation to compress data, enter the `WLST` script environment for Oracle Identity Federation and set the following properties:

- Set the `transientartifactrequestcompression` boolean property from the `datastore` group to `true` if the AuthnRequest for SSO Artifact profile should be compressed, otherwise set it to `false`:

  ```
  setConfigProperty('datastore',
  'transientartifactrequestcompression', 'true', 'boolean')
  ```

- Set the `transientartifactresponsecompression` boolean property from the `datastore` group to `true` if the assertion Response for SSO Artifact profile should be compressed, otherwise set it to `false`:

  ```
  setConfigProperty('datastore', 'transientartifactresponsecompression', 'true',
  'boolean')
  ```

- Set the `transientcompression` boolean property from the `datastore` group to `true` if the user session Data should be compressed, otherwise set it to `false`:

  ```
  setConfigProperty('datastore', 'transientcompression', 'true', 'boolean')
  ```

Note: Liberty 1.x support is deprecated.

6.6 Session Repository Configuration

This section contains topics related to maintaining the session repository.

6.6.1 Storing Assertion Attributes of User Session

The Oracle Identity Federation server features a session store containing the session information of the currently authenticated users. This session repository is capable of storing attributes that Oracle Identity Federation can use, when acting as identity provider (IdP), to populate SSO assertions.

The attributes stored in the user session can be added to the store in two ways:

- by a custom authentication engine, by setting a list of attributes to be saved in the user session
- with Oracle Identity Federation acting as a service provider (SP), when processing an incoming assertion; Oracle Identity Federation can save the attributes contained in the assertion, and the NameID and providerID in the user session

By default, for performance reasons, the storage of assertion information in the user session is disabled when Oracle Identity Federation acts as an SP.

To configure the Oracle Identity Federation server to store the assertion information, enter the `WLST` script environment for Oracle Identity Federation instance, and set the following property:

- Set the `sessionstoreassertionattrs` boolean property from the `spglobal` group to `true` if the attributes contained in the assertion, and the NameID and providerID, should be stored in the user session:

  ```
  setConfigProperty('spglobal',
  'sessionstoreassertionattrs', 'true', 'boolean')
  ```

- otherwise set it to `false`:

  ```
  setConfigProperty('spglobal',
  'sessionstoreassertionattrs', 'false', 'boolean')
  ```

6.7 Additional HTTP Configuration

This section contains additional topics for HTTP configuration for Oracle Identity Federation:

- Configuring HTTP-Only Flag for HTTP Cookies Set by Oracle Identity Federation
- Precautions when Customizing the Page in HTTP Post Profile
- Using a 303 Status Code for Redirects

6.7.1 Configuring HTTP-Only Flag for HTTP Cookies Set by Oracle Identity Federation

A non-standard extension to RFC2965 extends the `set-cookie` header further by specifying an `HttpOnly` flag. When you set this flag, the client (browser) should not make the cookie contents available to scripting environments. For example, the Java Script document.cookie method should not return the cookie contents. This significantly protects against "cross-site scripting" and similar attacks.

By default Oracle Identity Federation does not set the `HttpOnly` flag.

The Oracle Identity Federation server can be configured to set the `HttpOnly` flag when setting in the user's browser:

- the cookie used by Oracle Identity Federation to reference the user session
- the Oracle Access Manager cookie

To configure Oracle Identity Federation to set the `HttpOnly` header, enter the `WLST` script environment for Oracle Identity Federation and set the following properties:

1. Set the `cookiehttponlyenabled` boolean property from the `serverconfig` group to `true` if the `HttpOnly` flag should be set when sending the Oracle Identity Federation cookie to the browser, otherwise set it to `false`:

   ```
   setConfigProperty('serverconfig',
   'cookiehttponlyenabled', 'true', 'boolean')
   ```

2. Set the `oamcookiehttponlyenabled` boolean property from the `spengines` group to `true` if the `HttpOnly` flag should be set when sending the Oracle Access Manager cookie to the browser, otherwise set it to `false`:

   ```
   setConfigProperty('spengines',
   'oamcookiehttponlyenabled', 'true', 'boolean')
   ```

6.7.2 Precautions when Customizing the Page in HTTP Post Profile

The SAML/WS-Fed specifications define a POST profile where a SAML/WS-Federation server will redirect a user's browser to a remote SAML/WS-Fed implementation through the use of an HTML form.

Typically, such a server would send the browser an HTML page containing a FORM with:

- the action URL referencing the remote server
- some hidden fields containing SAML/WS-Fed message, and/or some attributes

When using that profile, the Oracle Identity Federation server prepares the action URL, the providerID referencing the remote server, and the list of hidden fields to send to the remove server. It then hands over this data to the `postprofile.jsp` page (contained in the `web.war` of the `$ORACLE_IDM_HOME/fed/install/oif.ear` file) that uses the information to build the HTML page to be presented to the browser.

You can customize that page to:

- Modify what is displayed to the browser
- Add extra fields to send to the remote server

Note: The remote SAML/WS-Federation server may not be able to process these fields, since they might not be compliant with the specifications.

When modifying the file to fit the particular needs of a deployment, be careful not to interfere with the POST profile, which can occur for example if you remove the required parameters/fields, such as the action URL or the hidden fields set by the Oracle Identity Federation server. To modify the file, unzip the `oif.ear` and the `web.war`, make the modification, and re-package the `web.war` and EAR file.

6.7.3 Using a 303 Status Code for Redirects

Oracle Identity Federation implements the SAML/WS-Fed/Liberty protocols that provide single sign-on (SSO) capabilities to HTTP clients, such as browsers. The protocols and profiles exercised at runtime during SSO operations can involve some HTTP redirects, where the Oracle Identity Federation server issues an HTTP redirect command to the browser.

Note: Liberty 1.x support is deprecated.

By default, Oracle Identity Federation uses the 302 HTTP status code when issuing a redirect. It is possible to configure the Oracle Identity Federation server to instead use a 303 HTTP status code when issuing a redirect provided the client supports HTTP 1.1.

To configure Oracle Identity Federation to use the 303 HTTP status code when possible, enter the `WLST` script environment for the Oracle Identity Federation instance, and set the following property:

- Set the `redirectuse302` boolean property from the `serverconfig` group to `false` if the Oracle Identity Federation server should use 303 HTTP status code when possible:

```
setConfigProperty('serverconfig',
'redirectuse302', 'false', 'boolean')
```

- otherwise set the property to `true`.

6.8 Additional Protocol Configuration

This section contains these topics:

- Configuring for eAuth Mode
- Configuring the BAE Direct Attribute Exchange Profile
- Configuring the SAML 2.0 LDAP Attribute Profile
- Configuring On-Demand Global Logout

6.8.1 Configuring for eAuth Mode

You can configure the Oracle Identity Federation server to comply with the eAuth specifications. Most of the configuration is performed through Fusion Middleware Control, but the specifications require the presence of two attributes in the SSO assertion that can only be configured through the MBeans/`WLST` scripts:

- the `us:gov:e-authentication:basic:specVer` attribute containing the version of the eAuth specifications supported by this server
- the `us:gov:e-authentication:basic:Sid` attribute containing the session identifier of the user performing the single sign-on

To configure Oracle Identity Federation to set those two attributes (for a specific provider) and to set the value of the eAuth version, enter the WLST script environment for Oracle Identity Federation instance, and set the following properties if needed:

- Set the `eauthmodeenabled` boolean property for the remote provider to `true` to enable the eAuth mode:

```
setFederationProperty(REMOTE_PROVIDER_ID,
'eauthmodeenabled', 'true', 'boolean')
##
## replace REMOTE_PROVIDER_ID with the identifier of the remote provider
```

- Set the `eauthversion` string property from the `idpglobal` group to the value the Oracle Identity Federation server should use (2.0 for example):

```
setConfigProperty('idpglobal', 'eauthversion', '2.0', 'string')
```

6.8.2 Configuring the BAE Direct Attribute Exchange Profile

As of 11g Release 1 (11.1.1.7.0), Oracle Identity Federation provides support for the Backend Attribute Exchange (BAE) Direct Attribute Exchange profile. This profile is based on the SAML Attribute Query profile defined by the SAML 2.0 specifications.

> **See Also:** "Security Markup Language (SAML) 2.0 Identifier and Protocol Profiles for Backend Attribute Exchange (BAE) v2.0" document at:
>
> `http://www.idmanagement.gov/sites/default/files/documents/BAE_v2_SAML2_Profile_Final_v1.0.0.pdf`

Configure Partners to Interact with BAE Direct Attribute Exchange Profile

To configure Oracle Identity Federation to interact with a partner using the BAE Direct Attribute Exchange profile:

- configure Oracle Identity Federation for Attribute Query protocol exchange with that partner.

  For more information, see:

  – Section 5.6, "Configuring Attribute Sharing with the Oracle Access Manager AuthZ Plug-in"

  – Section 5.8, "Web Services Interface for Attribute Sharing"

- Configure that partner for BAE mode. A boolean property must be set on that Oracle Identity Federation partner's entry so that Oracle Identity Federation uses the BAE profile rules when sending or processing a message for that partner.

Set the BAE Direct Attribute Exchange Profile for a Partner

To configure Oracle Identity Federation to set the BAE profile for a specific partner, enter the WLST script environment for the Oracle Identity Federation instance, and set the `attributebaeenabled` boolean property for the partner:

```
setFederationProperty("PARTNER_PROVIDER_ID", "attributebaeenabled" , "true",
"boolean")
```

to enable the BAE feature for that partner (replace *PARTNER_PROVIDER_ID* with the partner's ProviderID), and

```
setFederationProperty("PARTNER_PROVIDER_ID", "attributebaeenabled" , "false",
"boolean")
```

to disable the BAE feature for that partner (replace *PARTNER_PROVIDER_ID* with the partner's ProviderID).

6.8.3 Configuring the SAML 2.0 LDAP Attribute Profile

The SAML 2.0 specifications define the X.500 LDAP attribute profile, listing the attributes that an assertion must contain to be compliant with that profile.

The requirements are as follows:

- The format must be `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.
- The name must be a URI.
- The SAML Attribute element must specify an XML Encoding attribute and its value must be set to "LDAP".

The first two requirements are met by configuring the attribute for the Oracle Identity Federation server instance in Fusion Middleware Control.

The last requirement is met by configuring Oracle Identity Federation with WLST scripts; a property is set for a specific provider to which Oracle Identity Federation/IDP/AttributeAuthority will send the attributes contained in an assertion.

How to Use WLST for the X.500 LDAP Attribute Profile

Enter the WLST script environment for the Oracle Identity Federation server instance, then set the `attrx500ldapenabled` property for the remote provider to which Oracle Identity Federation will provide the assertion.

Set the `attrx500ldapenabled` boolean property to `true` to make the server compliant with the X.500 LDAP attribute profile. Otherwise set it to `false`:

```
setFederationProperty(REMOTE_PROVIDER_ID,
'attrx500ldapenabled', 'true', 'boolean')
##
## replace REMOTE_PROVIDER_ID with the identifier of the remote provider
```

6.8.4 Configuring On-Demand Global Logout

You can specify whether WS-Fed/SAML Global Logout should be executed when a logout operation is invoked at the Oracle Identity Federation server.

In a typical federation deployment, when the user invokes logout at the federation server, the flow is as follows:

- The user invokes the Oracle Identity Federation logout service at the `/fed/user/logout` URL.

  See Section 4.2.6, "Launch the Logout Process" for details about the logout service.

- Oracle Identity Federation:
  - logs the user out of the various authentication engines and SP integration modules (Oracle Access Manager, Oracle Single Sign-On, and others)
  - redirects the user for logout from the remote Federation partners involved in the current user session: this operation is called Global Logout.
  - finishes the logout operation once the global logout is complete.

You can disable the Global Logout flow with Fusion Middleware Control in two ways:

- globally, by selecting the "Local Logout Only" setting described in Section 5.2, "Configuring Server Properties".
- on a per-provider basis, by selecting the "Do not perform Global Logout with this Provider" setting on the Oracle Identity Federation Settings tab of the partner configuration section.

While these two approaches provide static control over the logout flow behavior, on-demand global logout lets you specify whether the user can invoke the global logout protocol at runtime.

To specify whether the user can choose global logout, you configure the federation server by setting the `slouserprefenabled` boolean property of the `serverconfig` group as follows:

- `true` to allow the user to choose global logout
- `false` to disallow the user from choosing global logout

To set the property, enter the WLST script environment for the Oracle Identity Federation server instance, and set the following property:

```
setConfigProperty('serverconfig', 'slouserprefenabled', 'true', 'boolean')
```

When on-demand global logout is enabled, the user can choose to perform the WS-Fed/SAML Logout operation by specifying the `globalslo` query parameter when invoking the Oracle Identity Federation logout service URL.

This parameter is of type boolean, and accepts one of two values:

- `true`, meaning that the global logout operation should be performed

- `false`, meaning that only the local logout should be performed

Following the instructions in Section 4.2.6, "Launch the Logout Process", the user invokes the service with a URL similar to:

```
http://hostname:port/fed/user/logout?returnurl=http%3A%2F%2Fanotherhostname%2Fpath
&globalslo=false
```

6.9 Protecting the SOAP Endpoint

Oracle Identity Federation provides two methods to protect the SOAP endpoint used in the SAML 1.x / SAML 2.0 / Liberty 1.x protocols:

- SSL with Client Authentication via SSL Certificate: the SOAP endpoint is protected with SSL, and by requiring an SSL Client certificate
- HTTP Basic Authentication: with this method, the SOAP endpoint is protected using the HTTP Basic Authentication mechanism.

Note: Liberty 1.x support is deprecated.

Topics include:

- SSL Client Authentication
- HTTP Basic Authentication

6.9.1 SSL Client Authentication

Refer to Section 8.1, "Configuring SSL for Oracle Identity Federation" for details on how to:

- configure SSL to protect the SOAP URL
- configure Oracle Identity Federation to connect to SOAP endpoints protected by SSL

6.9.2 HTTP Basic Authentication

This section describes:

- how to configure HTTP Basic Authentication on the server to protect SOAP URLs
- how to configure the credentials that are used when connecting to a remote server protected by HTTP basic authentication using the SOAP protocol

Note: When it is integrated with Oracle Single Sign-On with `mod_osso`, Oracle Identity Federation cannot be protected using HTTP Basic Authentication.

6.9.2.1 Configuring HTTP Basic Authentication to protect the SOAP URLs

This section lists the steps needed to protect the SOAP endpoints. The configuration changes are made on the Oracle WebLogic administration server.

The steps are as follows:

Configure Oracle WebLogic Server to check created policies

1. Log in to the Oracle WebLogic Server Administration Console.
2. On the left-hand pane, select **Security Realm**, and navigate to **myrealm**, then **Configuration**, then **Advanced**.
3. Select the following settings:
   - Check roles and Policies: "All Web applications and EJBs"
   - When Deploying Web Applications or EJBs: "Initialize roles and policies from DD"

     Click **Save**.
4. Stop the Administration server by navigating to **Environment**, then **Servers**, then **Control**, selecting "AdminServer" and clicking **Shutdown - Force Shutdown Now.**
5. From a terminal window, start the Administration server by invoking the script: `$DOMAIN_HOME/bin/startWebLogic.sh`.

Create a Group and a User

1. Log in to the Oracle WebLogic Server Administration Console.
2. On the left-hand pane, select **Security Realms** and navigate to **myrealm**, then **Users and Groups**, then **Groups**.
3. Click **New** and select a name (for example, soapusers). Click **OK**.
4. Navigate to **Users and Groups**, then **Users**.
5. Click **New** and select a name and password. Click **OK**.
6. Click on the user you just created and select the **Groups** tab.
7. Select the group you created and move it to the Chosen column. Click **Save**.

To enter additional users, repeat steps 4-7.

Create a Role and a Policy

1. Log in to the Oracle WebLogic Server's Administration Server console.
2. On the left-hand pane, select Deployments, expand the Oracle Identity Federation application, and click on "`/fed`".
3. Navigate to **Security**, then **Roles**.
4. Click **New** and select a name (for example, soapusers). In the URL Pattern, enter "`/`". Click **OK**.
5. Click on the role you just created and click **Add Conditions**.
6. Select Group and click **Next**.
7. Enter the name of the group you created and click **Add**, then **Finish**.
8. Click **Save**.
9. On the left-hand pane, select **Deployments**, expand the Oracle Identity Federation application and click on "`/fed`".
10. Navigate to **Security**, then **Policies**.
11. To protect the SOAP endpoint, you need to create a set of policies (one policy per URL you need to protect). The list of URLs that need to be protected is displayed in Table 6–2. To create a policy, follow these steps.:

a. Click **New** and enter the URL (from Table 1) that needs to be protected. Click **OK**.
b. Click on the policy you just created and click **Add Conditions**.
c. Select **Role** and click **Next**.
d. Enter the name of the role you created and click **Add**, then **Finish**.
e. Click **Save**.

12. After creating the group, users, role and policies, restart the administration and managed servers for the changes to take effect.

Table 6–2 URLs which Need Policies Created

Liberty 1.x/SAML 2.0 SOAP Endpoint	SAML 1.x SOAP Endpoint
/idp/soap	/idp/soapv11
/sp/soap	/sp/soapv11
/aa/soap	/aa/soapv11
/ar/soap	/authnauth/soapv11
/authnauth/soap	

6.9.2.2 Configuring Oracle Identity Federation to Connect to a Protected SOAP URL

On the client side, Oracle Identity Federation implements support for basic authentication when connecting to peer providers on the SOAP channel. You need to update the Oracle Identity Federation configuration so that the server can use the entered credentials when connecting to the SOAP endpoint of the remote provider.

Take these steps to enable Oracle Identity Federation to connect to a protected SOAP URL:

1. Log in to the Fusion Middleware Control console for the domain where Oracle Identity Federation is deployed.
2. Navigate to Oracle Identity Federation, then **Administration**, then **Federations**.
3. Select the remote provider that requires HTTP basic authentication on the SOAP channel, and click **Edit**.
4. In the Oracle Identity Federation Settings tab, select Enable HTTP Basic Authentication and enter the user name and password the server must use when connecting to the remote provider.
5. Click **Apply**.

6.10 Configuring the SAML 2.0 IdP Discovery (Common Domain Cookie) Profile

SAML 2.0 enables a service provider to discover the identity providers a user has used to authenticate. After authenticating a user, the IdP adds its Provider ID to the value of a cookie in the user's browser. The SP then reads this cookie and discovers the IdPs used. For the IdPs and SPs to write to and read from this cookie, the cookie must be in a domain common to all IdPs and SPs. Thus, this cookie is called the Common Domain Cookie (CDC).

When acting as an SP, if the CDC profile is enabled and an SSO operation is initiated without the provider ID of the target IdP, Oracle Identity Federation reads the common domain cookie and performs SSO with the first IdP in the list. You can also configure Oracle Identity Federation to prompt the user to choose the IdP with which to perform SSO. The user is then able to select from the list of all IdPs in the CDC that are also trusted by Oracle Identity Federation, acting as an SP.

This section describes how to configure Oracle Identity Federation to use the CDC profile. It contains these topics:

- Preliminary Steps to Set Up the CDC
- Configuring the CDC Profile as an Identity Provider
- Configuring the CDC Profile as a Service Provider
- Configuring Oracle Identity Federation to Display List of Trusted Providers in CDC

6.10.1 Preliminary Steps to Set Up the CDC

The common domain cookie is always marked as secure, so use of the CDC Profile requires enabling SSL, whether acting as an identity provider or a service provider. To enable SSL, follow the instructions in Section 8.1, "Configuring SSL for Oracle Identity Federation".

6.10.2 Configuring the CDC Profile as an Identity Provider

If Oracle Identity Federation is acting as an IdP, follow these steps to configure the CDC profile:

> **See Also:** Section 5.3, "Configuring Identity Providers - Common Properties"

1. Log in to Oracle Enterprise Manager and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Identity Provider**.
3. In the **Common** tab, check **Enable Common Domain** and enter the following properties:
   - Common Domain URL: The Oracle Identity Federation `intro` URL in the Common Domain where Oracle Identity Federation is listening:

     ```
     https://hostname_in_common_domain:sslport/fed/idp/intro
     ```

     For example:

     ```
     https://mycorp.commondomain.com:4443/fed/idp/intro
     ```

     > **Note:** This URL must use HTTPS and the SSL port that you configured earlier.

   - Name: The domain in which the cookie is written, for example,.commondomain.com.

Note: The name of the domain must always start with a leading period ".".

- Cookie Lifetime (day): The lifetime (in days) of the cookie

6.10.3 Configuring the CDC Profile as a Service Provider

If Oracle Identity Federation is acting as a service provider, follow these steps to configure the CDC profile:

See Also: Section 5.5, "Configuring Service Providers"

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Service Provider**.
3. In the **Common** tab, check **Enable Common Domain Cookie Service**, and enter the following property:
   - Service URL - The Oracle Identity Federation `introsso` URL in the Common Domain where Oracle Identity Federation is listening:

     ```
     https://hostname_in_common_domain:sslport/fed/sp/introsso
     ```

     For example:

     ```
     https://mycorp.commondomain.com:4443/fed/sp/introsso
     ```

Note: This URL must use HTTPS and the SSL port you configured earlier.

6.10.4 Configuring Oracle Identity Federation to Display List of Trusted Providers in CDC

Follow these steps to configure Oracle Identity Federation to prompt the user with the list of trusted IdPs in the common domain cookie when an SSO flow is initiated without the provider ID of the target IdP

1. Configure the CDC profile as described in Section 6.10.3, "Configuring the CDC Profile as a Service Provider"
2. Use the Oracle Identity Federation `WLST` commands or MBeans to set the `commondomainidpdiscenabled` property in Config "spglobal" to `true`.

Using the WLST Commands

Use the command:

```
setConfigProperty('spglobal', 'commondomainidpdiscenabled', 'true', 'BOOLEAN')
```

SeeChapter 9, "Oracle Identity Federation Command-Line Tools" for more information.

Using MBeans

In the ConfigMXBean named `spglobal`, invoke the `putProperty` operation with the following arguments:

- Name: "commondomainidpdiscenabled"
- Value: "true"
- Type: "BOOLEAN"

See Appendix A, "Oracle Identity Federation MBeans" for more information.

6.11 Configuring the Identity Provider Discovery Service

Identity provider discovery is a service that selects an identity provider (possibly through interaction with the user) to use during SSO. While Oracle Identity Federation does not provide an identity provider discovery service, it provides support for using such a service to select an IdP, if one is not passed in the authentication request to the SP during SP-initiated SSO.

For more information refer to the specifications at:

`http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-idp-discovery-cs-01.pdf`

If acting as a service provider, Oracle Identity Federation can be configured so that if an SSO operation is initiated without the provider ID of the target IdP, the user is redirected to a custom page to select the identity provider with which to perform SSO.

After the user selects an identity provider, the custom page resubmits the SSO request with the chosen IdP to Oracle Identity Federation.

Follow these steps to configure IdP discovery:

See Also: Section 5.5, "Configuring Service Providers"

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Service Provider**.
3. In the Common tab, check "Enable Identity Provider Discovery Service", and enter the following property:
   - `Service URL`: The location of the custom page displaying the IdP choices.

6.11.1 Create the IdP Discovery Service Page

Oracle Identity Federation redirects to the IdP Discovery Service page with the following parameters:

- `return`: This is the URL to which the page should send the new request containing the chosen IdP provider ID to Oracle Identity Federation.
- `returnIDParam`: This is the name of the parameter to use to specify the chosen IdP provider ID in the request sent to Oracle Identity Federation.

The page gets the value of these parameters, display a list of IdPs, and send a new request to Oracle Identity Federation specifying the chosen IdP Provider ID.

Note: Check that the URL query parameter values are correctly URL-encoded.

Example

The following is an example of an IdP discovery service page. This page allows the user to select an identity provider (from the list of provider IDs: `http://idp1.com`, `http://idp2.com`, `http://idp3.com`), and submit the chosen provider ID to Oracle Identity Federation to continue the SSO flow.

```
<%@ page buffer="5kb" autoFlush="true" session="false"%>
<%@ page language="java" import="java.util.*, java.net.*"%>

<%
// Set the Expires and Cache Control Headers
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

// Set request and response type
request.setCharacterEncoding("UTF-8");
response.setContentType("text/html; charset=UTF-8");

String submitURL = request.getParameter("return");
String returnIDParam = request.getParameter("returnIDParam");

List idps = new ArrayList();
idps.add("http://idp1.com");
idps.add("http://idp2.com");
idps.add("http://idp3.com");

%>

<html>
  <title>
  Select an Identity Provider
  </title>
<body bgcolor="#FFFFFF"><form  method="POST" action="<%=submitURL%>" id="PageForm"
name="PageForm" autocomplete="off">
    <center>
                <table cellspacing="2" cellpadding="5" border="0" width="500">
                    <tr><td colspan="2" align="center">
                         Select an Identity Provider
                    </td></tr>
                    </tr>
                    <tr>
                        <td align="right">Provider ID</td>
                        <td>
                           <select size="1" name="<%=returnIDParam%>">
<%
Iterator idpIT = idps.iterator();
while(idpIT.hasNext())
{
        String idp = (String)idpIT.next();
%>
                                <option value="<%=(idp)%>"><%=idp%></option>
<%
}
%>

                           </select>
                         </td>
                    </tr>
                    <tr>
```

```
                          <td colspan="2" align="center">
                             <input type="submit" value="Continue"/>
                          </td>
                     </tr>
                 </table>
       </center>
      </form>
     </body>
</html>
```

6.12 Setting up Infocard

Oracle Identity Federation can use Infocard as an authentication engine where the server acts as an RP (Resource Provider) for Infocard.

The flow is as follows:

- Oracle Identity Federation determines that the user needs to be challenged for authentication and selects the Infocard Authn Engine
- Oracle Identity Federation displays the login page containing the Infocard links. These links contain the claims that Oracle Identity Federation is requesting from the STS servers, and optionally the type of assertion to be returned. Oracle Identity Federation can also individually list the STS servers it recognized, or only one Infocard link, thus not listing the known STS servers.
- the user clicks on a link
- the Identity Selector installed on the user's machine is launched and displays the cards that can be used for this operation
- the user selects a card
- the Identity Selector connects to the STS server to retrieve a SAML assertion. (Note: the SAML assertion is encrypted using the Oracle Identity Federation SSL server certificate.)
- the Identity Selector presents the SAML assertion to Oracle Identity Federation
- Oracle Identity Federation decrypts the assertion using its encryption keystore, and validates the signature
- Oracle Identity Federation maps the SAML assertion to a local user, using the SAML 1.x or SAML 2.0 assertion mapping modules, based on the settings listed in the Service Provider page on the SAML 1.x and SAML 2.0 tabs. (Note: this step is similar to a Federation SSO flow, when Oracle Identity Federation acts as the service provider.)
- After mapping the assertion to a local user record, the authentication phase is complete and Oracle Identity Federation resumes the operation it was performing before the Infocard authentication engine was invoked

The Infocard authentication engine can additionally request optional attributes from the Infocard providers.

If the Infocard engine is instructed to return some attributes (requested by Oracle Identity Federation acting as IdP, because some of the assertion contents rely on user session attributes populated by the authentication engines), the engine adds those attributes as optional claims to be requested to the Infocard providers.

This section contains these topics:

- Server-side Infocard Setup
- Client-side Infocard Setup

6.12.1 Server-side Infocard Setup

Server-side setup includes the following:

- Set up JCE Policy Files for Oracle WebLogic Server
- Update the Oracle Identity Federation Configuration
- Add Personal Card Issuer STS
- Add Infocard Managed STS

6.12.1.1 Set up JCE Policy Files for Oracle WebLogic Server

Take these steps:

1. Download Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction policy files from this URL:

   `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

2. Unzip the files in all the `$JAVA_HOME/jre/lib/security` directories located under the Middleware home f older (to find those directories, look for `US_export_policy.jar` files). For every `$JAVA_HOME/jre/lib/security` directory, overwrite the default low strength `local_policy.jar` and `US_export_policy.jar` files with the ones provided by Oracle.

   See Also: What Is a Middleware Home? in the *Oracle Fusion Middleware Administrator's Guide*.

3. Restart the administration server and the managed server where Oracle Identity Federation is running.

6.12.1.2 Update the Oracle Identity Federation Configuration

Go to the Oracle Identity Federation instance in Fusion Middleware Control, and perform the following operations:

1. Infocard Authentication requires SSL. Configure SSL on Oracle WebLogic Server as explained in Section 8.1, "Configuring SSL for Oracle Identity Federation", and enable SSL on the Oracle Identity Federation server. You need to create a new SSL keystore if it does not already exist.

   The SSL server certificate must use the RSA public key algorithm since it is required for SAML encryption operations

2. For the Oracle Identity Federation encryption wallet, use the SSL keystore used for SSL support in Oracle WebLogic Server. When you use this keystore, the Infocard client uses the SSL server certificate to encrypt the assertion, thus requiring Oracle Identity Federation to use as the encryption wallet the key pair used for SSL traffic.

3. Load the SSL Java keystore as the Oracle Identity Federation encryption wallet. Navigate to **Administration**, then **Security and Trust**, and upload the new wallet keystore, specifying the password and alias.

Note: You must re-distribute the metadata to trusted partners since the encryption keystore was modified.

4. Navigate to **IdM Data Stores**, then **Authentication Engines**, and enable the Infocard engine. Depending on whether the engine should map the WS-Trust assertion to a record from the user data store, check or uncheck the box for **Map Assertion to User**.

Note: You can configure Oracle Identity Federation to add the authentication mechanism as a required claim, so that it is able to request a specific authentication method from the Infocard providers. To enable this feature, check the **Include Authentication Mechanism** box.

Note: Leave this box unchecked in most deployments unless Infocard providers support the feature.

If the box is checked, Oracle Identity Federation translates the local authentication mechanism values to SAML authentication methods, as defined in the authentication mechanisms mapping.

Finally, when the box is checked, Oracle Identity Federation verifies that a given Infocard provider supports the inclusion of the authentication mechanism by looking at the provider specific property called "Supports Authentication Mechanisms Claims," which is defined in the **Trusted Provider** settings of the Remote Provider configuration section.

Select Infocard as the default authentication engine if needed and save the changes.

If you check the **Map Assertion to User** box, the incoming assertion is mapped to a user record based on the configuration on the **SAML 2.0 / SAML 1.x Assertion** tab of the **Service Provider** page.

Note: When the Infocard authentication engine is invoked for authentication with the authentication mechanism that is mapped to the personal issuer card, only the personal issuer card is displayed on the login page. If invoked with an authentication mechanism different from the one mapped to the Personal Issuer Card, all the Infocard providers are displayed.

6.12.1.3 Add Personal Card Issuer STS

For Oracle Identity Federation to accept an assertion from the personal card issuer STS, it needs to have a trust relationship with the issuer. This trust is established by having the STS defined and enabled in the server's federations.

In Fusion Middleware Control, locate the Oracle Identity Federation instance and perform the following operations:

1. Navigate to **Administration**, then **Federations**, and add a WS-Fed 1.1 IdP identified by `http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self`

2. Select the STS, click **Update**, then select **Update Manually**.
3. From the SSO/Infocard Mode drop-down, select either **Infocard** if the STS only supports Infocard protocol, or **Single Sign-On and Infocard** if the STS supports Infocard and SSO protocols.
4. Infocard states that the relying party (Oracle Identity Federation in this case) should list the attributes or claims that the STS should include in the assertion it creates. With the attributes and the optional NameID contained in the assertion, Oracle Identity Federation can map the assertion to a local user record if configured for that operation.

   To add attributes to be requested for the STS, click **Attribute Mappings**.
5. Configure attribute mapping to list the attributes required by the Oracle Identity Federation server when the card selector is invoked; for each attribute marked "Require from Infocard", Oracle Identity Federation requires the attribute to be returned in the assertion from the WS-Trust server. The User Attribute Name is used to reference that attribute in Oracle Identity Federation, Assertion Attribute Name is the name of the attribute recognized by the STS, and Format/Namespace is the namespace to which the attribute is bound. The required claim from Oracle Identity Federation to the STS is the concatenation of the namespace, "/", and the assertion attribute name.

   For example:

   - Add an attribute entry User Attr Name=lastname, Assertion Attr Name=surname, Format or Namespace=http://schemas.xmlsoap.org/ws/2005/05/identity/claims. Check the **Require From Infocard** box.
   - Add another attribute entry with User Attr Name=firstname, Assertion Attr Name=givenname, Format or Namespace=http://schemas.xmlsoap.org/ws/2005/05/identity/claims. Check the **Require From Infocard** box.
6. Configure Oracle Identity Federation to map the assertion that the Personal Card Issuer will provide to a local user.

   For example, in the Oracle Identity Federation **Settings** tab, in the **Assertion Setting** tabs, uncheck the **Map User via NameID** box, check **Map User via Attribute Query** and enter the following LDAP query:
   `(&(sn=%lastname%)(givenname=%firstname%))`
7. Save the changes.
8. Check or uncheck the **Supports Authentication Mechanism Claims** box to indicate whether the authentication mechanism should be listed as a required Infocard attribute. Not all WS-Trust servers support the ability to specify the requested authentication mechanism through the use of attributes.
9. Save the changes.

6.12.1.4 Add Infocard Managed STS

For Oracle Identity Federation to accept an assertion from a remote STS, the Oracle Identity Federation server needs to have a trust relationship with the remote server. This trust is established by having the STS defined and enabled in the server's federations.

In Fusion Middleware Control, locate the Oracle Identity Federation instance and perform the following operations:

1. Add an entry by entering the STS provider ID, selecting IdP and the WS-Fed 1.1 version.
2. Select the STS, and click **Update**.
3. Enter the IdP signature verification certificate.
4. From the SSO/Infocard Mode drop-down, select either **Infocard** if the STS only supports Infocard protocol, or **Single Sign-On and Infocard** if the STS supports both Infocard and SSO protocols.
5. Infocard states that the relying party (Oracle Identity Federation in the present case) must list the attributes or claims that the STS should include in the assertion it creates. With the attributes and the optional NameID contained in the assertion, the Oracle Identity Federation server can map the assertion to a local user record (if configured for that operation).

   To add attributes to be requested for the STS, click **Attribute Mappings**.
6. Configure attribute mapping to list the attributes that the Oracle Identity Federation server will require when the card selector is invoked. For each attribute marked "Require from Infocard", Oracle Identity Federation requires the given attribute be returned in the assertion from the WS-Trust server. The User Attribute Name is used to reference that attribute in Oracle Identity Federation, Assertion Attribute Name is the name of the attribute recognized by the STS, and Format/Namespace is the namespace to which the attribute is bound. The required claim from Oracle Identity Federation to the STS is the concatenation of the Namespace, "/", and the assertion attribute name.

   For example:

   - Add an attribute entry with `User Attr Name=lastname, Assertion Attr Name=surname, Format` or `Namespace=http://schemas.xmlsoap.org/ws/2005/05/identity/claims`. Check the **Require From Infocard** box.
   - Add another attribute entry for `User Attr Name=firstname, Assertion Attr Name=givenname, Format` or `Namespace=http://schemas.xmlsoap.org/ws/2005/05/identity/claims`. Check the **Require From Infocard** box.
7. Configure Oracle Identity Federation to map the assertion that is provided by the Personal Card Issuer to a local user.

   For example, in the Oracle Identity Federation Settings tab, in the Assertion Setting tabs, uncheck the "Map User via NameID" box, check **Map User via Attribute Query** and enter the following LDAP query: `(&(sn=%lastname%)(givenname=%firstname%))`
8. Save the changes.

6.12.2 Client-side Infocard Setup

This section contains these topics:

- Import the Oracle Identity Federation SSL Certificate
- Create a Personal Infocard

6.12.2.1 Import the Oracle Identity Federation SSL Certificate

The client machine must trust the Oracle Identity Federation SSL certificate for Windows Cardspace to trust Oracle Identity Federation and allow the user to use

Infocards stored on the local computer. If the client does not trust the certificate authority that generated the SSL server, you must import the certificate.

Take these steps to import the certificate:

1. Using Internet Explorer, navigate to the URL with format `https://host:port`.
2. Right-click on the page.
3. Select **Properties**.
4. Select **Certificates**.
5. Click the **Certification Path** tab.
6. Select the CA that issued the certificate and view the certificate.
7. Click **Install Certificates**, and import the certificate in the trusted root Certification Authorities.

6.12.2.2 Create a Personal Infocard

Take these steps to create a personal Infocard with Windows Cardspace:

1. Go to the Windows control panel.
2. Double-click Windows Cardspace (if it is not present, install.NET from the Microsoft download site at `http://www.microsoft.com/downloads`).
3. Click **Add a Card**.
4. Select **Create a Personal Card** and fill in the fields.
5. Save the changes.

6.13 Additional Run-time Configuration

This section describes additional features you can configure to manage run-time behavior.

- Validating Target URLs for SSO and Logout Operations
- Providing XML Message to SP Engine after SSO Completes
- Customizing Error Pages
- Configuring Schema Validation for SSO Protocol Messages

6.13.1 Validating Target URLs for SSO and Logout Operations

When performing the SSO and Logout protocols, Oracle Identity Federation executes the SAML/WS-Fed protocol exchanges and then redirects the user to a final target URL, such as:

- a protected resource in case of SSO, or
- a returnurl when performing logout

These URLs can be specified as query parameters at runtime; for example, the `returnurl` query parameter for IdP-initiated SSO, logout flows, and so on.

Note: The `returnurl` query parameter value must be correctly URL Encoded

Here are some examples of flows where URLs can be specified:

- a user can start an IdP-initiated SSO flow by accessing:

  ```
  /fed/idp/initiatesso?providerid=SP_PROVIDER_ID&returnurl=http%3A%2F%2Furl.com
  ```

- a user can start the logout flow by accessing:

  ```
  /fed/user/logout?returnurl=http%3A%2F%2Furl.com
  ```

Oracle Identity Federation lets you validate URLs that can be specified at runtime. You configure validation specifying a list of approved hostnames, or approved domains.

> **Note:** A domain is a string beginning with '.', such as `.oracle.com` for the Oracle domain.

By default, validation is disabled.

To configure the return URL validation module, enter the `WLST` script environment for Oracle Identity Federation and set the `returnurlvalidationenabled` boolean property from the `serverconfig` group to `true` (or `false`) to enable (or disable) the module. For example:

```
setConfigProperty('serverconfig', 'returnurlvalidationenabled',
'true', 'boolean')
```

To add a host name or domain to the list of approved URLs/domains, enter the `WLST` script environment for Oracle Identity Federation, and issue these commands:

- Add a host name to the `returnurlvalidationlist` list:

  ```
  addConfigPropertyListEntry('serverconfig','returnurlvalidationlist',
  'hostname.domain.com','string')
  ```

- Add a domain to the `returnurlvalidationlist` list:

  ```
  addConfigPropertyListEntry('serverconfig','returnurlvalidationlist',
  '.domain.com','string')
  ```

6.13.2 Providing XML Message to SP Engine after SSO Completes

Oracle Identity Federation acting as SP can provide an XML message containing the assertion received by the server during the federated single sign-on flow. Depending on the binding used, this can be a SAML or SOAP message.

Note that:

- The XML message is provided to the SP engine with the attributes received in the assertion.
- The message is contained in the map referenced by `orafed-xmlmessage`.
- The attributes map is stored as an attribute in the `HttpServletRequest` object, referenced by `oracle.security.fed.sp.attributes`.

To enable sending the message, set the boolean property `spattrsincludexmlmessage` from the `spglobal` group to `true`.

To disable sending the message, set the property to `false`.

Note: `false` is the default configuration.

6.13.3 Customizing Error Pages

Errors can occur in Oracle Identity Federation for various reasons, such as:

- page not found
- federated single sign-on (SSO) error
- runtime error

When an error occurs, the server returns an error code (`404`, `401` or `500`) showing the Oracle WebLogic Server error page to the user.

You can configure Oracle Identity Federation to redirect the user to a custom page based on the error code.

Set the string property or `urlerror`*nnn* from the `serverconfig` configuration group to the URL to which the user should be redirected when Oracle Identity Federation returns the error, where *nnn* is 401, 404, or 500. (Thus, you can set the `urlerror401`, `urlerror404`, and `urlerror500` properties.)

Notes:

- `401` errors occur during Fed SSO operation if the federated SSO fails.
- `404` errors are raised when the user tries to access one of the Oracle Identity Federation servlets (`/fed/idp`, `/fed/sp`, `/fed/user`...) and the page is not found.
- `500` errors occur when fatal exceptions occur at runtime.
- If the server cannot initialize correctly, Oracle Identity Federation is unable to redirect the user to the `urlerror500` URL.

6.13.4 Configuring Schema Validation for SSO Protocol Messages

Oracle Identity Federation supports XML schema validation for SSO protocol messages.

This feature is implemented with the `schemavalidationenabled` property; validation is off by default.

To enable schema validation, enter the script environment for the Oracle Identity Federation server instance, and set the `schemavalidationenabled` property to `true`:

```
setConfigProperty('serverconfig','schemavalidationenabled','true','boolean')
```

To disable validation, set the property to `false` (default value).

```
setConfigProperty('serverconfig','schemavalidationenabled','false','boolean')
```

6.14 Additional Federation Data Store Configuration

When Oracle Identity Federation is configured to use an LDAP server or an RDBMS as its federation data store, the server performs various operations to create, locate, update, or delete federation records.

A federation record typically consists of the following data:

- IdP NameID: name identifier data created by the identity provider and used in the SAML messages
- SP NameID: name identifier data optionally set by the service provider during a Name Identifier Management update operation.

  If that NameID is set, it is used in SAML messages; otherwise, the IdP NameID is used.

During an operation that consumes an assertion, when Oracle Identity Federation acts as a service provider, the server tries to locate the federation record referenced in the NameID element contained in the assertion. By default, it first performs a lookup based on the SP NameID; if no results are returned, it performs a lookup based on the IdP NameID.

In some deployments, Oracle Identity Federation:

- might not be configured to do any NameID Management protocol exchanges, and
- might not have any of its federation records updated to set an SP NameID (that is, the administrator never performed an update operation on any federation records using the administrative tools)

In this case, the first federation record lookup performed during assertion consumption using the SP NameID will never return any records and serves to increase the response time.

If SP NameID lookup is not needed, it is possible to disable it to improve performance. To enable or disable the lookup, enter the `WLST` script environment for Oracle Identity Federation and make this configuration change:

Note: By default, the SP NameID lookup is enabled.

- Set the `fedusespnameidlookup` boolean property from the `datastore` group to `true` to enable the SP NameID lookup.
- Set the `fedusespnameidlookup` boolean property from the `datastore` group to `false` to disable the SP NameID lookup

For example:

```
setConfigProperty('datastore', 'fedusespnameidlookup', 'false', 'boolean')
```

6.15 Setting up Backwards Compatibility for Oracle Identity Federation 10g and ShareID service URLs

Background

Oracle Identity Federation 10*g*, and SHAREid/COREid Federation 2.x, provided service URLs for SAML 1.x and WS-Federation protocol support which were different from the SAML 2.0 and Liberty 1.x service URLs. These URLs have been modified in the 11g Oracle Identity Federation for consistency with the SAML 2.0 and Liberty 1.x service URLs. Customers upgrading to Oracle Identity Federation 11g, who use SAML 1.x or WS-Federation, must inform their partner providers of the new single sign-on service URLs.

Note: Liberty 1.x support is deprecated.

To ease that transition, Oracle Identity Federation 11g provides a separate module that allows backwards compatibility with the SHAREid service URLs. This module is a JavaEE application you can deploy alongside Oracle Identity Federation, to handle requests for the ShareID/Oracle Identity Federation 10g service URLs and redirect/forward them to the corresponding Oracle Identity Federation 11g service URLs.

Note: For the SAML1x protocol, in release 10*g* it was possible to set the authentication response profile binding by setting the "METHOD" query parameter to the desired profile while sending the authentication request. This feature is not supported in 11g, and you must configure the IdP to send the authentication response using the desired profile binding.

Procedure

Take these steps to set up a ShareID proxy:

Note: If a proxy is configured as in Appendix B, "Using Oracle HTTP Server as a Proxy for Oracle Identity Federation", you need to modify the oif.conf file to also divert /shareid URLs to Oracle WebLogic Server.

See Also: Getting Started with Oracle WebLogic Server Administration Console in the *Oracle Fusion Middleware Administrator's Guide*.

1. At the Oracle WebLogic Server Administration Console, navigate to the Deployments page.
2. Click **Lock & Edit**.
3. Click **Install**.
4. Navigate to the location of the shareidupdate.ear file (located in `$ORACLE_HOME/fed/install`).
5. Click **Next**.
6. Select **Install this deployment as an application** and click **Next**.
7. Select the name of the managed server and click **Next**.
8. Select **Advanced** in the **Security** section.
9. Click **Finish**.
10. Click **Apply Changes**.
11. Returning to the **Deployments** page, find the new application called "shareidupdate" (Note: you may have to click **Next** if the application does not appear in the first 10 entries).
12. If the state of the shareidupdate application is not listed as "Active" select the application and click **Start**, servicing all requests, and click **yes**. Now locate the

shareidupdate application in the Deployments page; its state should be listed as "Active".

By default, the `shareidupdate` proxy uses the incoming protocol (HTTP or HTTPS), server name, and server port as the protocol, server name, and server port to which messages should be redirected. Consider using non-default values (when using a proxy server, for example).

To set these values:

- Find the `shareidupdate` `web.xml` file; it should be in a subfolder of:

  ```
  DOMAIN_HOME/servers/SERVER_NAME/tmp/_WL_user/shareidupdate
  ```

- Add the properties `IsSecure`, `ServerName`, and/or `ServerPort` to the servlets being used. (Note: `IsSecure` is set to `true` if desired protocol is HTTPS, `false` if the desired protocol is HTTP.)
- Save changes and restart the application.

For example, the `<servlet>` element may now contains elements such as:

```
<init-param>
      <param-name>IsSecure</param-name>
      <param-value>true</param-value>
</init-param>
<init-param>
      <param-name>ServerPort</param-name>
      <param-value>7777</param-value>
</init-value>
```

6.16 Mapping Users through Attributes and NameID in SP Mode

Oracle Identity Federation acting as an SP can locate a user based on the attributes and name identifier value stored in an assertion without using any federation records.

When configured not to use the federated identity to map the assertion to a user record, Oracle Identity Federation/SP uses the NameID and the attributes contained in the incoming assertion to map the user in the repository. Once the user record is located, Oracle Identity Federation/SP creates an authenticated session for that user in the identity and access management framework and redirects the user to the final target URL.

This flow does not use any federation records, so it is not necessary to have a federation data store configured to use Oracle Identity Federation as the service provider.

If Oracle Identity Federation cannot locate the user during the flow, the default behavior is to return a `401 Unauthorized` error to the user. You can configure Oracle Identity Federation to redirect the user to the authentication engine instead, so that custom corrective measures such as user account provisioning can be initiated. This behavior is implemented with the **Error when User Mapping fails** property; see Section 6.16.2, "Configuring Oracle Identity Federation", under the procedure titled "If the Mapping Fails."

On returning to Oracle Identity Federation from the authentication engine, if the user still cannot be mapped, a final result of `401` is returned.

Limitations

Note these limitations:

- Since Oracle Identity Federation/SP does not store any federation records when configured to map the assertion without using federated identities, no account linking information is available in the Identity Federation section of Fusion Middleware Control.
- Additionally, the Name Identifier Update and Federation Termination profiles will not complete; if the peer IdP sends a message for one of these profiles, Oracle Identity Federation/SP will return an error message indicating that the federation record could not be found.

This section contains these topics:

6.16.1 Locating a User

You have two options for locating a user record in the repository:

- Using the Name ID Format mapping, where the NameID is linked to a user attribute.
- Using an LDAP/SQL query that involves the NameID and the attributes stored in the assertion.

If both options are enabled, Oracle Identity Federation/SP first uses the NameID mapping search, and if no results are returned, it uses the LDAP/SQL query flow.

The query contains placeholders that are replaced by the attribute and NameID values contained in the assertion. The placeholders use a `%NAME%` format in which Oracle Identity Federation/SP replaces `NAME` with:

- An attribute name, referencing an attribute contained in the assertion. When creating the query, Oracle Identity Federation/SP replaces the %AttributeName% with the value of the attribute referenced by AttributeName.

  Note: The attribute mapping module will have mapped the attributes, contained in the assertion, to the attribute name/values configured for the remote provider. The attribute name needs to reference an attribute from this list.

- orafed-nameid-value - indicates that this placeholder should be replaced by the Name ID value
- orafed-nameid-qualifier - indicates that this placeholder should be replaced by the Name ID qualifier

- orafed-nameid-format - indicates that this placeholder should be replaced by the Name ID format
- orafed-providerid - indicates that this placeholder should be replaced by the Peer ProviderID

6.16.2 Configuring Oracle Identity Federation

The SAML 2.0 module supports the use of federated identities, but not the SAML 1.x modules.

To configure Oracle Identity Federation to use federated identities for assertion to user mapping operations:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Check or uncheck **Map User via Federated Identity**.

To map a user using the NameID:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Check **Map User via NameID**.
4. Configure the NameID format and the attribute in the user record to be used during the lookup procedure.

To map a user using an LDAP/RDBMS query:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Check **Map User via Attribute Query**.
4. Enter the LDAP or SQL query to be used during the lookup procedure.

If the Mapping Fails

If the mapping fails, you can configure Oracle Identity Federation to invoke the authentication engine instead of returning a 401-error code. To configure the server, perform the following steps:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Uncheck **Error when User Mapping fails** to invoke the authentication engine that will then have access to the content of the assertion and its attributes.

Note: The attribute mapping module will have mapped the attributes, contained in the assertion, to the attribute name/values configured for the remote provider. The attribute name needs to reference an attribute from this list.

6.16.3 Example 1: Assertion Mapping without federated identities using NameID for SAML 2.0

In this example, Oracle Identity Federation/SP uses the NameID contained in the assertion to look up a local user in the LDAP user data store. The format of the NameID is `emailAddress`, and the search uses the mail attribute of the LDAP user record.

The server is configured to use the NameID mapping functionality to locate the user. Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Uncheck **Map User via Federated Identity**.
4. Uncheck **Map User via Attribute Query**.
5. Check **Map User via NameID**.
6. Enable **Email Address NameID Format**, and enter the attribute of the user record holding the email address (mail typically for LDAP server).
7. Check **Error when User Mapping fails**; this will force Oracle Identity Federation to return a `401` error to the browser if the user cannot be located.
8. Apply the changes.

6.16.4 Example 2: Simple Assertion Mapping without Federated Identities with an LDAP/SQL Query

In this example, Oracle Identity Federation/SP uses the NameID contained in the assertion to look up a local user in the LDAP user data store. The format of the NameID is `emailAddress`, and the search uses the mail attribute of the LDAP user record.

The server is configured to use the LDAP/SQL Query functionality to locate the user. Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Uncheck **Map User via Federated Identity**.
4. Check **Map User via Attribute Query**.
5. Enter the following LDAP query in the Attribute Query field:
   `(&(mail=%orafed-nameid-value%))`
6. Uncheck **Map User via NameID**.
7. Check **Error when User Mapping fails**; this forces Oracle Identity Federation to return a `401` error to the browser if the user cannot be located.
8. Apply the changes.

6.16.5 Example 3: Complex Assertion Mapping without Federated Identities with an LDAP/SQL Query

In this example, Oracle Identity Federation/SP uses the SAML attributes for email address and last name in the assertion to look up a local user in the LDAP user data store.

The `mail` and `sn` local attributes are obtained from the LDAP user record. The attributes in the assertion are referenced as email and lastname. Oracle Identity Federation/SP is not configured for attribute name mapping, so the LDAP query uses the attribute names contained in the SAML assertion; if attribute name mapping was configured, the LDAP query would use the names resulting from the attribute name mapping (refer to Section 5.9, "Configuring Attribute Mapping and Filtering" for more information).

The server is configured to use the LDAP/SQL query functionality to locate the user. Perform the following configuration steps:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Uncheck **Map User via Federated Identity**.
4. Check **Map User via Attribute Query.**
5. Enter the following LDAP query in the Attribute Query field:
   `(&(mail=%email%)(sn=%lastname%))`
6. Uncheck **Map User via NameID**.
7. Check **Error when User Mapping fails**; this forces Oracle Identity Federation to return a `401` error to the browser if the user cannot be located.
8. Apply the changes.

6.16.6 Example 4: Assertion Mapping without Federated Identities using LDAP/SQL Query and NameID Mapping

In this example, Oracle Identity Federation/SP uses the email address contained in the NameID to locate the user. If the operation fails, the last name SAML attribute from the assertion is used to look up a local user in the LDAP user data store, using the local attributes `mail` and `sn` from the LDAP user record.

The server is configured to use both NameID Mapping and LDAP/SQL Query to locate the user. Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Uncheck **Map User via Federated Identity**.
4. Check **Map User via Attribute Query**.
5. Enter the following LDAP query in the Attribute Query field: `(&(sn=%lastname%))`
6. Check **Map User via NameID.**
7. Enable **Email Address NameID Format**, and enter the attribute of the user record holding the email address (mail typically for LDAP server).

8. Check **Error when User Mapping fails**; this forces Oracle Identity Federation to return a `401` error to the browser if the user cannot be located.
9. Apply the changes.

6.16.7 Example 5: Assertion Mapping without Federated Identities for a Specific IdP

If Oracle Identity Federation/SP needs an attribute-based authentication configuration specific to a peer identity provider, the setup information must be stored in the IdP's entry in the Federations list.

In this example, Oracle Identity Federation /SP is set up for attribute-based authentication for an IdP referenced by `http://idp.com`. Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Federations**.
3. Select the identity provider and click **Update**.
4. Click the **Oracle Identity Federation Settings** tab.
5. Expand the **Service Provider/Requester Settings** section, and go to assertion settings.
6. Uncheck **Map User via Federated Identity.**
7. Check **Map User via Attribute Query**.
8. Enter the following LDAP query in the Attribute Query field:
   `(&(mail=%email%)(sn=%lastname%))`
9. Uncheck **Map User via NameID**.
10. Check **Error when User Mapping fails**; this forces Oracle Identity Federation to return a `401` error to the browser if the user cannot be located.
11. Apply the changes.

6.17 Automatic Account Linking Based on Attribute Query Mapping

Automatic account linking at the SP allows the service provider to directly map an identity contained in an assertion to a user.

When Oracle Identity Federation is acting as a service provider, and is configured to use federated identities to map the incoming SAML 2.0 assertion, it can automatically create a federation record by locating a user based on the attributes and name identifier received in an assertion.

This section contains topics related to account linking:

- Locating the User
- Configuring Oracle Identity Federation
- Example 1: Automatic Account Linking through NameID mapping for SAML 2.0
- Example 2: Simple Automatic Account Linking through LDAP/SQL Query
- Example 3: Complex Automatic Account Linking through LDAP/SQLQuery
- Example 4: Automatic Account Linking through LDAP/SQL Query and NameID Mapping
- Example 5: Automatic Account Linking via Attribute Query for a Specific IdP

6.17.1 Locating the User

When configured to use federated identities and Automatic Account Linking is enabled, the administrator has two options for locating a user record in the repository:

- Using the Name ID Format mapping, where the NameID is linked to a user attribute. This uses the existing mapping.
- Using an LDAP/SQL query that involves the NameID and the attributes stored in the assertion.

If both options are enabled, Oracle Identity Federation/SP first uses the NameID mapping search, and if no results are returned, it uses the LDAP/SQL query flow. If Oracle Identity Federation/SP cannot locate the user record during this flow, the server challenges the user for credentials.

The administrator specifies in Oracle Identity Federation configuration the LDAP/SQL query to be used when trying to look up a user. The query contains placeholders that are replaced by the attribute and NameID values contained in the assertion. The placeholders use a `%NAME%` format in which Oracle Identity Federation/SP replaces `NAME` with:

- An attribute name, referencing an attribute contained in the assertion. When creating the query, Oracle Identity Federation/SP replaces `%AttributeName%` with the value of the attribute referenced by `AttributeName`.

Note: The attribute mapping module maps the attributes contained in the assertion to the attribute name/values configured for the remote provider. The attribute name needs to reference an attribute from the list.

- orafed-nameid-value - Oracle Identity Federation replaces this placeholder with the Name ID value
- orafed-nameid-qualifier - Oracle Identity Federation replaces this placeholder with the Name ID qualifier
- orafed-nameid-format - Oracle Identity Federation replaces this placeholder with the Name ID format
- orafed-providerid - Oracle Identity Federation replaces this placeholder with the Peer ProviderID

6.17.2 Configuring Oracle Identity Federation

Only the SAML 2.0 module supports the use of federated identities, not the SAML 1.x modules.

To configure Oracle Identity Federation to use federated identities for assertion to user mapping, and to enable automatic account linking operations:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Check **Map User via Federated Identity**.
4. Check **Enable Auto Account Linking**.

To map a user using the NameID:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Check **Map User via NameID**.
4. Configure the NameID Format enabled and the attribute in the user record to be used during the lookup procedure of the automatic account linking operation.

To map a user using an LDAP/RDBMS query:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Check **Map User via Attribute Query**.
4. Enter the LDAP or SQL query to be used during the lookup procedure of the automatic account linking operation.

6.17.3 Example 1: Automatic Account Linking through NameID mapping for SAML 2.0

In this example, Oracle Identity Federation/SP uses the NameID contained in the assertion to look up a local user in the LDAP user data store. The format of the NameID is `emailAddress`, and the search uses the mail attribute of the LDAP user record.

The server is configured to use the NameID mapping functionality to locate the user during automatic account linking.

Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Check **Map User via Federated Identity**.
4. Check **Map Enable Auto Account Linking.**
5. Uncheck **Map User via Attribute Query**.
6. Check **Map User via NameID**.
7. Enable **Email Address NameID Format**, and enter the attribute of the user record holding the email address (mail typically for LDAP server).
8. Apply the changes.

6.17.4 Example 2: Simple Automatic Account Linking through LDAP/SQL Query

In this example, Oracle Identity Federation/SP uses the NameID contained in the assertion to look up a local user in the LDAP user data store and automatically create the federation record. The format of the NameID is `emailAddress`, and the search uses the mail attribute of the LDAP user record.

The server is configured to use the LDAP/SQL query functionality to locate the user. Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.

3. Check **Map User via Federated Identity**.
4. Check **Map Enable Auto Account Linking**.
5. Check **Map User via Attribute Query**.
6. Set the attribute query to `(&(mail=%orafed-nameid-value%))`
7. Uncheck **Map User via NameID**.
8. Apply the changes.

6.17.5 Example 3: Complex Automatic Account Linking through LDAP/SQLQuery

In this example, Oracle Identity Federation/SP uses the email address and the last name SAML attributes in the assertion to look up a local user in the LDAP user data store and automatically creates the federation record. The local attributes `mail` and `sn` from the LDAP user record are used. The attributes in the assertion are referenced as email and lastname.

Oracle Identity Federation/SP is not configured for attribute name mapping, so the LDAP query uses the attribute names contained in the SAML assertion; if attribute name mapping were configured, the LDAP query would use the names resulting from attribute name mapping (refer to Section 5.9, "Configuring Attribute Mapping and Filtering" for more information on Attribute Name Mapping).

The server is configured to use the LDAP/SQL query functionality to locate the user. Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.
3. Check **Map User via Federated Identity.**
4. Check **Map Enable Auto Account Linking**.
5. Check **Map User via Attribute Query**.
6. Set the Attribute Query to `(&(mail=%email%)(sn=%lastname%))`.
7. Uncheck **Map User via NameID**.
8. Apply the changes.

6.17.6 Example 4: Automatic Account Linking through LDAP/SQL Query and NameID Mapping

In this example, Oracle Identity Federation/SP uses the email address contained in the NameID to locate the user to create the federation record. If the operation fails, Oracle Identity Federation/SP uses the last name SAML attribute from the assertion to look up a local user in the LDAP user data store. The local attributes mail and sn from the LDAP user record are used.

The server is configured to use the NameID Mapping and LDAP/SQL Query features to locate the user.

Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **SAML 2.0**, then **Assertion Settings**.

3. Check **Map User via Federated Identity**.
4. Check **Map Enable Auto Account Linking**.
5. Check **Map User via Attribute Query**.
6. Set the Attribute Query to `(&(sn=%lastname%))`.
7. Check **Map User via NameID**.
8. Enable **Email Address NameID Format**, and enter the attribute of the user record holding the email address (mail typically for LDAP server).

6.17.7 Example 5: Automatic Account Linking via Attribute Query for a Specific IdP

If Oracle Identity Federation/SP needs an attribute-based authentication configuration specific to a peer identity provider, then the setup information needs to be stored in the IdP's entry in the Federations list.

In this example, Oracle Identity Federation/SP is using federated identities and is set up for automatic account linking through attribute query for an IdP referenced by http://idp.com.

Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Federations**.
3. Select the identity provider and click **Update**.
4. Click the **Oracle Identity Federation Settings** tab.
5. Expand the service provider/Requester Settings section, and go to assertion settings.
6. Check **Map User via Federated Identity**.
7. Check **Map Enable Auto Account Linking**.
8. Check **Map User via Attribute Query**.
9. Enter the following LDAP query in the Attribute Query field: `(&(mail=%email%)(sn=%lastname%))`.
10. Uncheck **Map User via NameID**.
11. Check **Error when User Mapping fails**; this forces Oracle Identity Federation to return a `401` error to the browser if the user cannot be located.
12. Apply the changes.

6.18 User Opt-In and Opt-Out for Single Sign-On

You can configure Oracle Identity Federation IdP to determine if a user has given (or denied) permission to perform federated single sign-on for the user, based on the value of an attribute in the user's directory record.

If consent has been given, SSO operations can be performed automatically if the user is authenticated at Oracle Identity Federation/IdP, or within the identity and access management (IAM) framework integrated with Oracle Identity Federation. If consent has not been obtained, Oracle Identity Federation/IdP must challenge the user for credentials every time a Federation SSO operation occurs, even if the user is already authenticated at Oracle Identity Federation in the IAM domain.

Note: In this section, Oracle Identity Federation/IdP refers to Oracle Identity Federation acting as identity provider.

Topics in this section include:

6.18.1 Modes of Operation

Oracle Identity Federation/IdP can implement this feature in three modes:

1. Off - The Opt-in/Opt-out functionality is not exercised
2. Opt-In - If the user attribute for opt-in/opt-out equals the value set by the administrator, Oracle Identity Federation/IdP does not force the user to re-authenticate for Federation SSO operations; otherwise it forces re-authentication.
3. Opt-Out - If the user attribute for opt-in/opt-out equals the value set by the administrator, then Oracle Identity Federation/IdP forces the user to re-authenticate for Federation SSO operations; otherwise it does not force re-authentication.

6.18.2 Configuring Oracle Identity Federation

To configure Oracle Identity Federation to use Opt-In/Opt-Out:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Identity Provider**.
3. Select the **Opt-In/Opt-Out** mode:
   - Off: indicates that the **Opt-in/Opt-out** feature is not exercised
   - Opt-In: indicates that the Opt-in mode is active
   - Opt-Out: indicates that the Opt-out mode is active
4. If the mode is set to Opt-In or Opt-Out, then enter the Opt-In/Out user attribute that references the attribute to retrieve from the user record. Its value is compared against the value set by the administrator.
5. If the mode is set to Opt-In or Opt-Out, then enter the Opt-In/Out attribute value holding the value set by the administrator and used to compare against the user attribute.

6.18.3 Example 1: Off Mode

In this example, the opt-in/opt-out feature is turned off so that the user is never re-challenged for credentials when a federation record is created on Oracle Identity Federation/IdP.

Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Identity Provider**.
3. Select Off as the Opt-In/Opt-Out mode.
4. Apply the changes.

6.18.4 Example 2: Opt-In Mode

In this example, the opt-in/opt-out feature is set to Opt-In, the attribute containing the user setting is fedrecordcreation, and the value indicating that the user opted in is `agreed`.

Oracle Identity Federation/IdP re-challenges the user for credentials during a federation creation operation only if the `fedrecordcreation` attribute value of the user is different from `agreed`.

Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Identity Provider**.
3. Select Opt-In as the **Opt-In/Opt-Out** mode.
4. Set the **Opt-In/Out User Attribute** to `fedrecordcreation`.
5. Set the **Opt-In/Out Attribute Value** to `agreed`.
6. Apply the changes.

6.18.5 Example 3: Opt-Out Mode

In this example, the feature is set to `optout`, the attribute containing the user setting is `fedrecordcreation` and the value indicating that the user opted in is `disallowed`.

Oracle Identity Federation/IdP re-challenges the user for credentials during a federation creation operation only if the user's `fedrecordcreation` attribute value equals `disallowed`.

Perform the following steps to configure Oracle Identity Federation/SP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Identity Provider**.
3. Select Opt-Out as the **Opt-In/Opt-Out** mode.
4. Set the **Opt-In/Out User Attribute** to `fedrecordcreation`.
5. Set the **Opt-In/Out Attribute Value** to `disallowed`.
6. Apply the changes.

6.18.6 Example 4: Opt-In Mode for a Specific IdP

If Oracle Identity Federation/IdP needs an Opt-In mode configuration specific to a peer service provider, then the setup information needs to be stored in the SP's entry in the Federations list.

In this example, the opt-in/opt-out feature is set to Opt-In, the attribute containing the user setting is `fedrecordcreation`, and the value indicating that the user opted in is `agreed`, for an SP referenced by `http://sp.com`.

Oracle Identity Federation/IdP re-challenges the user for credentials during a federation creation operation only if the `fedrecordcreation` attribute value of the user is different from `agreed`.

Perform the following steps to configure Oracle Identity Federation/IdP:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Federations**.
3. Select the service provider and click **Update**.
4. Click the **Oracle Identity Federation Settings** tab.
5. Expand the **Identity Provider/Authority Settings** section.
6. Select Opt-In as the **Opt-In/Opt-Out** mode.
7. Set the **Opt-In/Out User Attribute** to `fedrecordcreation`.
8. Set the **Opt-In/Out Attribute Value** to `agreed`.
9. Apply the changes.

6.19 Bypassing User Mapping During Assertion Processing

With this feature Oracle Identity Federation, when acting as a service provider, does not attempt to locate a user based on the information contained in the assertion; instead the content of the assertion is passed directly back to the SP Integration module, which implements the user mapping flow.

If Oracle Identity Federation/SP is configured to bypass mapping (that is, to not map the principal identified in the assertion to a local user), Oracle Identity Federation does the following:

- creates an Oracle Identity Federation session for the anonymous user, specified in the Oracle Identity Federation administration console in the service provider section. This is required as the server needs to be aware of the user being authenticated at the server and at peer providers (for example, at the logout operations).

  Thus, setting the Anonymous User ID in the Oracle Identity Federation pages for Fusion Middleware Control is mandatory.
- passes the NameID, attributes, and other information back to the SP Integration module, as specified in Section 10.4.2.3, "Implementing the Service", under the heading "Oracle Identity Federation Assertion Processing".

6.19.1 Configuring Oracle Identity Federation

To configure Oracle Identity Federation to map (or not map) the incoming assertion to a user record:

1. Log in to Fusion Middleware Control.
2. Navigate to **Administration**, then **Service Provider**, then **Common**.
3. Check **Map assertion to User Account** to configure Oracle Identity Federation to map incoming assertions to user records; uncheck it to not map the assertion.
4. Apply the changes.

6.20 Overriding NameID Mapping Per Partner

On a per-partner basis, an Oracle Identity Federation administrator can override the mapping of NameID formats to local user directory attributes.

To configure this feature at the command line take these steps:

1. Set up the script environment as described in Chapter 9, "Oracle Identity Federation Command-Line Tools."
2. Invoke the WLST shell using the appropriate command for your platform.

   Windows

   ```
   IDM_ORACLE_HOME\common\bin\wlst.cmd
   ```

   Linux

   ```
   IDM_ORACLE_HOME/common/bin/wlst.sh
   ```

3. Enter one of these commands depending on the name format used:

   ```
   setFederationProperty('SPproviderID','nameformatemail','attribute-name','st
   ring')
   ```

   if name format is Email Address

   ```
   setFederationProperty('SPproviderID','nameformatx500','attribute-name','str
   ing')
   ```

   if name format is X509 Subject Name

   ```
   setFederationProperty('SPproviderID','nameformatunspecified','attribute-nam
   e','string')
   ```

   if name format is Unspecified

   ```
   setFederationProperty('SPproviderID','nameformatkerberos','attribute-name',
   'string')
   ```

   if name format is Kerberos

   ```
   setFederationProperty('providerID','nameformatwindows','attribute-name','st
   ring')
   ```

   if name format is Windows Domain Qualified Name

   ```
   setFederationProperty('providerID','nameformatcustom','attribute-name','str
   ing')
   ```

   if name format is Custom

   Note: If federation store was set and a federation record exists for the user, the nameid in the federation record is used.

6.21 Configuring Audience Restrictions for Assertions

When using assertions to exchange information, SAML authorities such as an identity provider or attribute authority can set the conditions under which an assertion is valid. Typical conditions might be:

- Time before which the assertion is not valid

- Time after which the assertion is not considered valid any more
- List of providers that can process the assertion. Only a provider listed in the `AudienceRestictionCondition` element of the assertion is able to use the assertion.

The SAML specifications define the `AudienceRestictionCondition` as a list of `Audience` elements, each one referencing a provider that can process the assertion.

By default, Oracle Identity Federation creates an `AudienceRestrictionCondition` element when generating an assertion, and includes the recipient of the assertion using these rules:

- For SAML 1.x protocol exchanges, set the Audience as the Assertion Consumer Service URL of the service provider.
- For SAML 2.0 protocol exchanges, set the Audience as the `ProviderID` of the service provider /Attribute Requestor.
- For WS-Fed protocol exchanges using SAML assertions, set the Audience as the `ProviderID` of the service provider.

When Oracle Identity Federation receives and processes an assertion, by default it validates the `AudienceRestrictionCondition`, if present, by using the `ProviderID` or URL where the assertion was posted.

Depending on the deployment scenario, it might be necessary to disable generation and validation of the `AudienceRestrictionCondition` element; you can do so either at a protocol level (SAML 1.0, SAML 1.1 or SAML 2.0 assertions), or at the trusted provider level.

To configure Oracle Identity Federation to control generation and processing of the `AudienceRestrictionCondition` for SAML 1.x/SAML 2.0 assertions at a global level, enter the `WLST` script environment for the Oracle Identity Federation instance, and:

- Set the `audiencerestrictionenabled` boolean property from the `idpsaml10`, `idpsaml11` or `idpsaml20` groups to `true` (default) to enable the generation of `AudienceRestrictionCondition` when creating a SAML 1.0, SAML 1.1 or SAML 2.0 assertion respectively.

  ```
  setConfigProperty('idpsaml11', 'audiencerestrictionenabled', 'true', 'boolean')
  ```

  Set it to `false` to disable the generation of the condition
- Set the `audiencerestrictionenabled` boolean property from the `spsaml10`, `spsaml11` or `spsaml20` groups to `true` (default) to enable the validation of `AudienceRestrictionCondition` when processing a SAML 1.0, SAML 1.1 or SAML 2.0 assertion respectively:

  ```
  setConfigProperty('spsaml11', 'audiencerestrictionenabled', 'true', 'boolean')
  ```

  Set it to `false` to disable validation of the condition.

To configure Oracle Identity Federation to enable generation and processing of the `AudienceRestrictionCondition` for a specific trusted provider, enter the `WLST` script environment for the Oracle Identity Federation instance, and set the `audiencerestrictionenabled` boolean property for a trusted provider referenced by `REMOTE_PROVIDER_ID` to `true`:

```
setFederationProperty(REMOTE_PROVIDER_ID, 'audiencerestrictionenabled', 'true',
'boolean')
```

Set the property to `false` to disable generation and processing of the condition.

You can also configure Oracle Identity Federation to use a custom string when:

- Oracle Identity Federation/IdP creates an assertion.

  Oracle Identity Federation uses the custom string specified in the configuration to populate the `AudienceRestrictionCondition` element

- Oracle Identity Federation/SP processes an assertion.

  Oracle Identity Federation validates the `AudienceRestrictionCondition` element, if present, by comparing it to the custom string specified in the configuration.

To configure Oracle Identity Federation to use a specific audience value when validating the `AudienceRestrictionCondition` for SAML 1.x/SAML 2.0 assertions at a global level, enter the `WLST` script environment for Oracle Identity Federation instance, and set the `audiencerestrictionvalue` string property from the `spsaml10`, `spsaml11` or `spsaml20` groups to the custom value that Oracle Identity Federation uses during the validation of `AudienceRestrictionCondition` when processing a SAML 1.0, SAML 1.1 or SAML 2.0 assertion respectively:

```
setConfigProperty('spsaml11', 'audiencerestrictionvalue', 'someglobalvalue',
'string')
```

If you set the `audiencerestrictionvalue` to the empty string value, Oracle Identity Federation/SP validates the `AudienceRestrictionCondition` element as shown above.

To configure Oracle Identity Federation to use a specific Audience value when generating/validating the `AudienceRestrictionCondition` for a specific trusted provider, enter the `WLST` script environment for Oracle Identity Federation instance, and set the `audiencerestrictionvalue` string property for a trusted provider referenced by `REMOTE_PROVIDER_ID` to use a custom string to generate and validate the condition when creating and processing an assertion:

```
setFederationProperty(REMOTE_PROVIDER_ID, 'audiencerestrictionvalue',
'customvalue', 'string')
```

If you set the `audiencerestrictionvalue` to the empty string value, Oracle Identity Federation/SP populates/validates the `AudienceRestrictionCondition` element as shown above.

6.22 Certificate Path Validation

Oracle Identity Federation provides a certificate validation module (described in Section 5.10.3, "Security and Trust - Trusted CAs and CRLs") that validates any certificate used for XML digital signature verification by using the certificates of the Trusted CAs and the CRLs uploaded by the administrator.

The module integrates with the JRE CertPathValidation API to validate certificates using the default CertPathValidation module configured in the JVM. When the default CertPathValidation module is the Sun implementation, Oracle Identity Federation can leverage the Online Certificate Status Protocol (OCSP) and the CRL Distribution Point (CDP) features provided by the Sun module.

You manage the certificate validation flow using the following properties:

- In Fusion Middleware Control, navigate to the Oracle Identity Federation server instance, then Security and Trust, then Trusted CAs and CRLs section:
  - Checking the "Enable Certificate Validation" box enables certificate validation in Oracle Identity Federation

  - The Trusted Certificate Authorities table lists all the known and trusted certificates of the CAs
  - The Certificate Revocation Lists table contains the CRLs used to check the revocation status of certificates.
- the certpathvalidationenabled boolean property in the serverconfig configuration group determines the validation module to be used:
  - `false` means that Oracle Identity Federation's internal certificate validation module is used, based on the Trusted Certificate Authorities and Certificate Revocation Lists tables.
  - `true` means that the JRE CertPathValidation module is used. This module is bootstrapped with the contents of the Trusted Certificate Authorities table to serve as the basis for the trusted CAs.
- the `certpathvalidationcrlenabled` boolean property in the `serverconfig` configuration group indicates whether the JRE CertPathValidation module should check for certificate revocation using the CRLs listed in the Certificate Revocation Lists table:
  - `false` disables CRL validation in the JRE CertPathValidation module
  - `true` enables CRL validation in the JRE CertPathValidation module
- the `certpathvalidationocspenabled` boolean property in the `serverconfig` configuration group indicates whether the OCSP plugin of the Sun CertPathValidation implementation is enabled.
  - false disables OCSP validation
  - true enables OCSP validation

Notes: ■The `certpathvalidationcrlenabled` must be set to `true` to enable the OCSP plugin.

- Sun CertPathValidation module performs OCSP validation first and, if the operation is inconclusive, it performs a CRL revocation check operation.

- the `certpathvalidationocspurl` string property in the `serverconfig` configuration group contains the URL of the OCSP server where the Sun CertPathValidation module sends the request for validation.

Note: (To enable the OCSP functionality set the `certpathvalidationocspenabled` boolean property to `true`.

- the `certpathvalidationocspcertsubject` string property in the `serverconfig` configuration group contains the subject name of the OCSP server's certificate.

Notes:

- The matching certificate must also be added to the Oracle Identity Federation trusted CA certificate store.
- Ensure the OCSP functionality is enabled with the certpathvalidationocspenabled boolean property set to true.

- the `certpathvalidationcdpenabled` boolean property in the `serverconfig` configuration group indicates whether the Sun CertPathValidation module will use CDP extensions when performing a CRL revocation check operation.

If the OCSP module is enabled, Oracle Identity Federation sets the following Java Security properties in the JVM:

- `ocsp.enable` is set to `true`
- `ocsp.responderURL` is set to the OCSP server's URL
- `ocsp.responderCertSubjectName` is set to the subject name of the OCSP server's certificate

See Also: The Java PKI Programmers Guide for more detail on these system properties:

`http://java.sun.com/javase/6/docs/technotes/guides/security/certpath/CertPathProgGuide.html#AppC`

If the CDP functionality is enabled, the `com.sun.security.enableCRLDP` Java system property is set to `true`.

Note: For the OCSP and CDP features to be enabled, it is important to set the default CertPathValidation module to the Sun implementation (true by default in a standard installation).

6.23 Sending the ACS URL with the Authentication Request

In some cases, the SAML 2.0 IdP will require the SP to send the Assertion Consumer Service (ACS) URL as part of the authentication request.

This feature enables you to configure Oracle Identity Federation to send the ACS URL in the SAML 2.0 authentication request, referencing the location where the IdP should redirect the user to complete the Federation SSO operation. By default, the feature is set to `false` (do not send the ACS URL).

You can configure this feature at the SAML 2.0 global level or SAML 2.0 partner level using the WLST command.

To configure this feature, enter the WLST script environment for Oracle Identity Federation and set the following properties:

- To configure the server to send the ACS URL in the Authn Request at the SAML 2.0 Global level, set the sendacsurlauthnrequest boolean property from the spsaml20 group to `true`; to not send the URL, set it to `false`. For example:

```
setConfigProperty('spsaml20', 'sendacsurlauthnrequest', 'true', 'boolean')
```

- To configure the server to send the ACS URL in the Authn Request for a specific IdP partner, set the sendacsurlauthnrequest boolean property for that partner to `true`; to not send the URL to that partner, set it to `false`. For example:

```
setFederationProperty(REMOTE_IDP_PROVIDER_ID, 'sendacsurlauthnrequest',
'true', 'boolean')
```

6.24 Integrating with an OpenID Partner

This section explains how to configure Oracle Identity Federation to work with OpenID partners and relying partners. Topics include:

- Integrating with an OpenID Provider (OP)
- Integrating with a Relying Party (RP)
- Configuring Attributes

6.24.1 Integrating with an OpenID Provider (OP)

To integrate with an OpenID provider, you must provision the partner information in Oracle Identity Federation, and provide certain information to the federated partner.

6.24.1.1 Provision the OP Partner

Take these steps to provision the OP partner:

1. Create a partner by locating the Oracle Identity Federation server in Fusion Middleware Control, navigating to the Federations page and specifying these elements:
   - OP's ProviderID: This value is used when starting a Federation SSO flow with that OP, for example with the `/fed/sp/initiatesso` URL. If Oracle Identity Federation is the OP, enter a URL like `http(s)://hostname:port/fed/idp/openidv20`.
   - Protocol Version: OpenID 2.0
   - Role: Identity Provider
2. Edit the partner with the following settings in the Trusted Provider Settings tab:
   - Discovery URL: This is the URL where the OP publishes its XRDS metadata. If Oracle Identity Federation is the OP, use a URL like `http(s)://hostname:port/fed/idp/openidv20`.
   - Endpoint URL: This is the URL where the user is redirected at the OP for authentication. If Oracle Identity Federation is the OP, use a URL like http(s)://hostname:port/fed/idp/openidv20.

6.24.1.2 Provide Data to the OP

To integrate with an OP, Oracle Identity Federation/RP may need to provide some information to the federated partner. The OP may need the following data:

- Realm: This is the URL which identifies the RP. The Realm URL for Oracle Identity Federation/RP is a URL like `http(s)://hostname:port/fed/sp/openidv20`, and it also acts as the Oracle Identity Federation/RP OpenID ProviderID.
- Endpoint URL: This is the URL to which the user is redirected from the OP to Oracle Identity Federation/RP with the authentication information. The Endpoint URL for Oracle Identity Federation/RP is a URL like `http(s)://hostname:port/fed/sp/openidv20`.

6.24.2 Integrating with a Relying Party (RP)

To integrate with a relying party, you must provision the RP partner, and provide certain information to the RP.

6.24.2.1 Provision the RP Partner

If the Generic OpenID SP Provider was not created with the OpenID 2.0 tab of the IdP page in Fusion Middleware Control to allow any RP to perform federated SSO with Oracle Identity Federation/OP, you must provision the RP partner.

1. Create a partner on the Federations page by specifying:
   - The RP's ProviderID: This is the RP realm value and is used when starting a Federation SSO flow with that RP (for example through the `/fed/idp/initiatesso` URL). It is also used for RP discovery by downloading the RP XRDS metadata from the Realm URL. If Oracle Identity Federation is the RP, enter a URL like `http(s)://hostname:port/fed/sp/openidv20`.
   - Protocol Version: OpenID 2.0
   - Role: Service Provider
2. Edit the partner to set the following on the Trusted Provider Settings tab:
   - Endpoint URL: This is the URL to which the user is redirected from the OP to Oracle Identity Federation/RP with the authentication information. If Oracle Identity Federation is the RP, enter a URL like `http(s)://hostname:port/fed/sp/openidv20`.

6.24.2.2 Provide Data to the RP

To integrate with an RP, the Oracle Identity Federation/OP may need to provide information to the federated partner. The RP may require the following data:

- Discovery URL: This is the URL where Oracle Identity Federation/OP publishes its XRDS metadata. The Discovery URL for OIF/OP is a URL like `http(s)://hostname:port/fed/idp/openidv20`.
- Endpoint URL: This is the URL to which the user is redirected from the RP to Oracle Identity Federation/OP for authentication. The Endpoint URL for Oracle Identity Federation/OP is a URL like `http(s)://hostname:port/fed/idp/openidv20`.

6.24.3 Configuring Attributes

Attributes must be configured to enable Oracle Identity Federation to work with OpenID partners.

6.24.3.1 Attributes for Oracle Identity Federation as an OP

To enable Oracle Identity Federation/OP to send attributes in OpenID responses,enable the OpenID Attibute Exchange (AX) protocol as follows:

1. In Fusion Middleware Control, navigate to the Identity Provider page and select OpenID 2.0 tab.
2. Check the "Enable Attribute Exchange (AX) 1.0" box.
3. Save the changes.

For each RP (or for the Generic OpenID SP Provider), you must configure Oracle Identity Federation/OP to send attributes for that RP, and define the attributes that can be included in the OpenID response:

1. In Fusion Middleware Control, navigate to the Federation page.
2. Select the RP and click **Edit**.

3. Click "Attribute Mappings and Filters".
4. Add an entry to represent an attribute that can be sent in an OpenID response.
5. Click **Add**.
6. Enter the name of the attribute from the user data store in the "User Attribute Name" field.
7. The name of the attribute in the OpenID response consists of the "Format or Namespace" and the "Assertion Attribute Name", where "Format or Namespace" contains the URL prefix of the attribute name, and "Assertion Attribute Name" contains the suffix. For example, defining the http://openid.net/schema/contact/internet/email attribute involves setting the "Format or Namespace" to `http://openid.net/schema/contact/internet/` and the "Assertion Attribute Name" to `email`.
8. Check the "Get Value from User Session" box if the attribute should be retrieved from the Oracle Identity Federation user session.
9. Check the "Send with SSO Assertion" box if this attribute should always be sent in the OpenID response, even if the RP did not request it.
10. Click **OK**.
11. After adding all attributes, click **OK**.
12. Check the "Enable Attributes in Single Sign-On (SSO)" box.
13. Click **Apply**.

6.24.3.2 Attributes for Oracle Identity Federation as an RP

Oracle Identity Federation/RP can request attributes at runtime through a custom pre-SP actions module or a custom SP engine, or based on configuration for the OP that will authenticate the user.

To configure Oracle Identity Federation/RP to request attributes from an OP:

1. In Fusion Middleware Control, navigate to the Federation page.
2. Select the OP and click **Edit**.
3. Click "Attribute Mappings and Filters".
4. Add an entry to represent an attribute that will be requested from the OP.
5. Click **Add**.
6. Enter the name to be locally used to represent this attribute in the "User Attribute Name" field.
7. The name of the attribute in the OpenID response consists of the "Format or Namespace" and the "Assertion Attribute Name", where "Format or Namespace" contains the URL prefix of the attribute name, and "Assertion Attribute Name" contains the suffix. For example, defining the http://openid.net/schema/contact/internet/email attribute involves setting "Format or Namespace" to `http://openid.net/schema/contact/internet/` and "Assertion Attribute Name" to `email`.
8. Check the "Require from Infocard" box to request this attribute from this OP.
9. Click **OK**.
10. After adding all attributes, click **OK**.

6.25 Implementing the OpenID UI Extension

Oracle Identity Federation supports the OpenID User Interface Extension 1.0 (UI Ext), which defines a mechanism to support OpenID user interfaces optimized for different environments and languages.

You can instruct Oracle Identity Federation to include data in the redirect URL to the IdP indicating how the IdP should interact visually with the user.

The configuration parameters are as follows:

`openiduiextenabled (boolean, false by default)`
Configures Oracle Identity Federation/IdP to publish that it supports UI Ext. Intended only for testing Oracle Identity Federation as an IdP.

`openiduiextlangenabled (boolean, false by default)`
Configures Oracle Identity Federation/SP to send the UI Ext lang parameter in the redirect URL from Oracle Identity Federation/SP to IdP, if the IdP supports UI Ext Lang.

`openiduiextpopupenabled (boolean, false by default)`
Configures Oracle Identity Federation/SP to send the UI Ext mode parameter set to popup in the redirect URL from Oracle Identity Federation/SP to IdP, if the IdP supports UI Ext mode.

`openiduiexticonenabled (boolean, false by default)`
Configures Oracle Identity Federation/SP to publish in its XRDS it supports UI Ext icon, with the value specified in the `openiduiexticonurl` parameter shown below.

`openiduiexticonurl (string, empty by default)`
Contains the URL location of the icon to be used by the IdP. Will be published in the SP XRDS .

To configure this feature, enter the WLST script environment for Oracle Identity Federation and set the following properties:

```
setConfigProperty('idpopenid20', 'openiduiextenabled', 'true', 'boolean')
setConfigProperty('spopenid20', 'openiduiextlangenabled', 'true', 'boolean')
setConfigProperty('spopenid20', 'openiduiextpopupenabled', 'true', 'boolean')
setConfigProperty('spopenid20', 'openiduiexticonenabled', 'true', 'boolean')
setConfigProperty('spopenid20', 'openiduiexticonurl', 'http://my.icon.com',
'string')
```

Note: Replace *http://my.icon.com* with the icon's URL location.

7 Diagnostics and Auditing

This chapter describes Oracle Enterprise Manager Fusion Middleware Control monitoring and logging features for Oracle Identity Federation. It contains these sections:

- Monitoring
- Availability
- Logging
- Auditing

Note: Liberty 1.x support is deprecated.

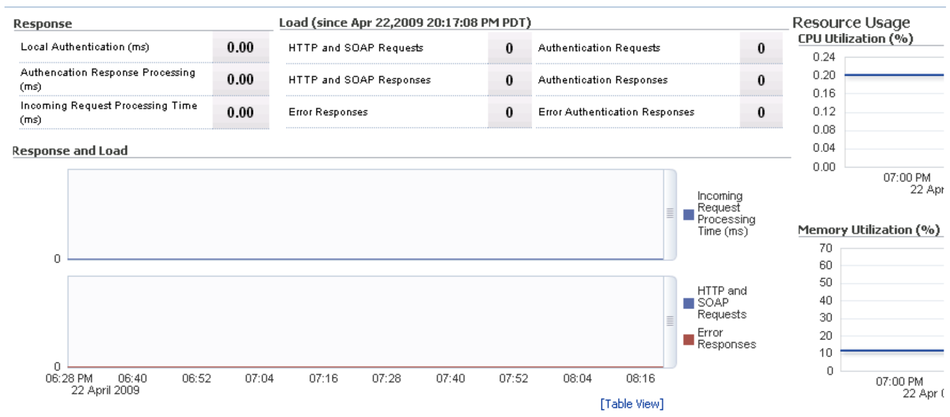
7.1 Monitoring

This section describes how to monitor your Oracle Identity Federation server.

- Oracle Identity Federation Home Page
- Performance Summary

7.1.1 Oracle Identity Federation Home Page

This is the home page for your Oracle Identity Federation server instance.



Response

Local Authentication (ms)	0.00
Authencation Response Processing (ms)	0.00
Incoming Request Processing Time (ms)	0.00

Load (since Apr 22,2009 20:17:08 PM PDT)

HTTP and SOAP Requests	0	Authentication Requests	0
HTTP and SOAP Responses	0	Authentication Responses	0
Error Responses	0	Error Authentication Responses	0

This page summarizes statistics about the server instance. For details about the metrics shown here, see Section 7.1.2, "Performance Summary".

7.1.2 Performance Summary

Oracle Identity Federation provides a number of built-in metrics to enable application developers, system administrators, and others to measure application-specific performance information. Metrics for system, state, and phase events are available in these functional areas:

- Protocol Profiles
- Enterprise Data Tier Connectivity
- Security Protocol Messages
- Data Model (JVT DiscoveryProviders)

This section contains these topics:

- About Sensor Weights
- Event Metrics
- State Events
- Phase Events

7.1.2.1 About Sensor Weights

The DMS sensor weight is a setting on the managed server on which Oracle Identity Federation is running; the sensor weight determines which metrics you see:

- all - all sensors are activated.
- normal (or no weight value set) - all the sensors at the normal level are activated
- heavy - all the sensors at the default level and at the heavy level are activated
- None - none of the sensors is activated.

Given the cost of running expensive instrumentation, setting the sensor weight to conditionally activate only the necessary sensors lets you efficiently collect relevant metric data about the server.

Set the Sensor Weight

If you start Oracle WebLogic Server using the administration console, set the `-Doracle.dms.sensors=`*`level`* property in the `servers/`*`serverName`*`/server start/arguments` section of the server, where *`level`* is one of the sensor levels described above.

If you start Oracle WebLogic Server through a script, set the `-Doracle.dms.sensors=`*`level`* property in the *`domain_home`*`/bin/startManagedWebLogic.sh` script.

7.1.2.2 Event Metrics

This section contains these topics:

- Protocol Profiles
- Security Processing

Note: In the table, the **Label and Description** refers to the short label attached to the metric in Fusion Middleware Control, followed by a description of the metric.

7.1.2.2.1 Protocol Profiles Table 7–1 shows the protocol profile metrics:

Table 7–1 Protocol Profile Events

Name	Label, Description	Weight
Requests	HTTP and SOAP Requests Total number of requests received. This is the addition of the RequestsHTTP and RequestsSOAP request messages.	normal
RequestsHTTPRedirect	HTTP Requests using Redirect Binding Total number of requests sent or received using HTTP Redirect binding.	normal
RequestsHTTPPOST	HTTP Requests using POST Binding Total number of requests sent or received using HTTP-POST binding.	normal
RequestsHTTPPOSTSimpleSign	HTTP Requests using POST Simple Sign Binding Total number of requests sent or received using HTTP-POST Simple Sign binding.	normal
RequestsSOAP	SOAP Requests Total number of requests sent or received using the SOAP binding.	normal
WellFormedRequests	Received XML Requests successfully parsed Total number of well-formed requests received, that is, those that resulted in no XML translation errors.	normal
BadlyFormedRequests	Received XML Requests with parsing failures Total number of badly formed requests, that is, those that resulted in XML translation errors.	normal
SignedRequests	Requests signed Total number of requests sent or received with message level signatures.	normal
EncryptedRequests	Requests encrypted Total number of requests sent or received with message level encryption.	normal
SignedAndEncryptedRequests	Requests both signed and encrypted Total number of requests sent or received with message level signatures and encryption.	normal
Responses	HTTP or SOAP Responses Total number of responses sent or received. This is the sum of the ResponsesHTTP and ResponsesSOAP response messages.	normal
ResponsesHTTPRedirect	HTTP Responses using Redirect Binding Total number of responses sent or received using HTTP Redirect binding.	normal
ResponsesHTTPPOST	HTTP Responses using POST Binding Total number of responses sent or received using HTTP-POST binding.	normal
ResponsesHTTPPOSTSimpleSign	HTTP Responses using POST Simple Sign Binding Total number of responses sent or received using HTTP-POST Simple Sign binding.	normal

Table 7–1 (Cont.) Protocol Profile Events

Name	Label, Description	Weight
ResponsesSOAP	SOAP Responses Total number of responses sent or received using the SOAP binding.	normal
ErrorResponses	Error Responses Total number of responses sent or received with error status	normal
SignedResponses	Responses Signed Total number of responses sent or received with message level signatures	normal
EncryptedResponses	Responses Encrypted Total number of responses sent or received with message level encryption	normal
SignedAndEncryptedResponses	Response both Signed and Encrypted Total number of responses sent or received with message level signatures and encryption	normal
AttributeQueryRequests	AttributeQuery Requests Total number of `<AttributeQuery>` requests sent by the SP or received by the IDP.	normal
AttributeQueryResponses	AttributeQuery Responses Total number of `<Response>` responses sent by the IDP or received by the SP.	normal
AttributeQueryErrorResponses	AttributeQuery Error Responses Total number of `<Response>` error responses sent by the IDP or received by the SP.	normal
AuthnRequestRequests	AuthnRequest Requests Total number of `<AuthnRequest| Request>` requests sent by the SP or received by the IDP.	normal
AuthnRequestResponses	AuthnRequest Responses Total number of `<Response|AuthnResponse>` responses sent by the IDP or received by the SP.	normal
AuthnRequestErrorResponses	AuthnRequest Error Responses Total number of `<Response| AuthnResponse >` error responses sent by the IDP or received by the SP.	normal
SecurityTokenResponses	RequestSecurityToken Responses Total number of `<RequestSecurityTokenResponse >` responses sent by the IDP or received by the SP.	normal
LogoutRequests	Logout Requests Total number of `<LogoutRequest| SignOut>` requests sent or received.	normal
LogoutResponses	Logout Responses Number of LogoutResponse messages sent or received.	normal
LogoutErrorResponses	Logout Error Responses Number of LogoutResponse messages with error status sent or received.	normal

Table 7–1 (Cont.) Protocol Profile Events

Name	Label, Description	Weight
NameIDManagementRequests	ManageNameIDRequests Total number of `<ManageNameIDRequest|RegisterNameIdentifier|FederationTerminationNotification>` requests sent or received.	normal
NameIDManagementResponses	ManageNameIDResponses Total number of `<ManageNameIDResponse>` or RegisterNameIdentifier responses sent or received.	normal
NameIDManagementErrorResponses	ManageNameID Error Responses Total number of `<ManageNameIDRequest|RegisterNameIdentifier|FederationTerminationNotification>` error responses sent or received.	normal
ArtifactResolutionRequests	ArtifactResolve Requests Total number of `<ArtifactResolve|Request>` requests sent by the SP or received by the IDP.	normal
ArtifactResolutionResponses	ArtifactResolve Responses Total number of `<ArtifactResponse|Response>` responses sent by the IDP or received by the SP.	normal
ArtifactResolutionErrorResponses	ArtifactResolve Error Responses Total number of `<ArtifactResponse|Response>` error responses sent by the IDP or received by the SP.	normal
NameIdentifierFormat_Persistent	NameIDs of Persistent format processed Total usage of Persistent Name Identifier in messages processed at the SP or IDP.	normal
NameIdentifierFormat_Transient	NameIDs of Transient format processed Total usage of Transient Name Identifier in messages processed at the SP or IDP.	normal
NameIdentifierFormat_Unspecified	NameIDs of Unspecified format processed Total usage of Unspecified Name Identifier in messages processed at the SP or IDP.	normal
NameIdentifierFormat_EmailAddress	NameIDs of EmailAddress format processed Total usage of Email Address Name Identifier in messages processed at the SP or IDP.	normal
NameIdentifierFormat_X509DN	NameIDs of X509SubjectName format processed Total usage of X.509 Subject Name Identifier in messages processed at the SP or IDP.	normal
NameIdentifierFormat_Windows	NameIDs of WindowsDomainQualifiedName format processed Total usage of Windows Domain Qualified Name Identifier in messages processed at the SP or IDP.	normal
NameIdentifierFormat_Kerberos	NameIDs of Kerberos format processed Total usage of Kerberos Principal Name Identifier in messages processed at the SP or IDP.	normal
RequestProcessed	ApplicationController Requests Total number of requests processed by ApplicationController.	normal

7.1.2.2.2 Security Processing

Table 7–2 shows the protocol profile metrics:

> **Note:** In the table, the **Label and Description** refers to the short label attached to the metric in Fusion Middleware Control, followed by a description of the metric.

Table 7–2 Security Processing Events

Name	Label, Description	Weight
XMLSignatures_Signed	XML Signatures Generated Total number of XML signatures generated.	normal
XMLSignatures_Verified	XML Signatures Verification Successes Total number of XML signatures verified successfully.	normal
XMLSignatures_VerifyFailed	XML Signatures Verification Failures Total number of XML signature verification failures.	normal
XMLEncryption_Encryptions	XML Encryptions Generated Total number of XML encryptions generated.	normal
XMLEncryption _Decryptions	XML Decryption Successes Total number of successful XML decryptions.	normal
XMLEncryption _ DecryptionFailures	XML Decryption Failures Total number of XML decryption failures.	normal

7.1.2.3 State Events

The following metric is collected for enterprise data-tier connectivity:

- Server_OpenSessions - The Fusion Middleware Control label for this metric is "Open Server Connections". It represents the total number of open connections with LDAP or RDBMS server.

  The sensor weight is "all".

7.1.2.4 Phase Events

This section contains these topics:

- Data Model
- Protocol Profiles
- Security Processing

7.1.2.4.1 Data Model

Table 7–3 shows the JVTDiscoveryProviders metrics by phase event sensors:

> **Note:** In the table, the **Label and Description** refers to the short label attached to the metric in Fusion Middleware Control, followed by a description of the metric.

Table 7–3 JVTDiscoveryProvider Events

Name	Label and Description	Weight
ArtifactCreation	SAML Artifact Creation Time (ms) Time taken to create the artifact by Artifact DiscoveryProvider.	heavy
LocateArtifact	SAML Artifact Retrieval Time (ms) Time taken to locate the artifact by Artifact DiscoveryProvider.	heavy
LocateConfiguration	Server Configuration Retrieval Time (ms) Time taken to locate protocol/server configuration by Configuration DiscoveryProvider.	heavy
LocateMetadata	Provider Metadata Retrieval Time (ms) Time taken to locate the metadata by Metadata Discovery Provider.	heavy
ProfileStateCreation	ProfileState Object Creation Time (ms) Time taken to create profile state by ProfileState DiscoveryProvider.	heavy
LocateProfileState	ProfileState Object Retrieval Time (ms) Time taken to locate profile state by ProfileState DiscoveryProvider.	heavy
SessionCreation	User Session Retrieval or Creation Time (ms) Time taken to create or locate the user session by Session DiscoveryProvider.	heavy
LocateUser	User Object Retrieval Time (ms) Time taken to locate the user by User DiscoveryProvider.	heavy
LocateSession	Session Object Retrieval Time (ms) Time taken to locate the session.	heavy
CreateActiveServiceProviderFederation	Active SP Federation Creation Time (ms) Time taken to create the active service provider federation.	heavy
LocateActiveServiceProviderFederation	Active SP Federation Retrieval Time (ms) Time taken to locate the active service provider federation.	heavy
CreateActiveIdentityProviderFederation	Active IdP Federation Creation Time (ms) Time taken to create the active Identity Provider federation.	heavy
LocateActiveIdentityProviderFederation	Active IdP Federation Retrieval Time (ms) Time taken to locate the active Identity Provider federation.	heavy
LocateProviderFederation	Provider Federation Retrieval Time (ms) Time taken to locate the Provider federation.	heavy
LocateTemporaryProviderFederation	Temporary Provider Federation Retrieval Time (ms) Time taken to locate the active Temporary Provider federation	heavy
CreateAffiliationProviderFederation	Affiliation Federation Creation Time (ms) Time taken to create the Affiliation Provider federation.	heavy

Table 7–3 (Cont.) JVTDiscoveryProvider Events

Name	Label and Description	Weight
LocateAffiliationFederation	Affiliation Federation Retrieval Time (ms) Time taken to locate the Affiliation federation	heavy
CreateServiceProviderFederation	SP Federation Creation Time (ms) Time taken to create the service provider federation	heavy
CreateIdentityProviderFederation	IdP Federation Creation Time (ms) Time taken to create the Identity Provider federation.	heavy
DeleteSession	Session Deletion Time (ms) Time taken to delete the session.	heavy
CreateBinaryLargeObject	Database BLOB Creation Time (ms) Time taken to create the Blob.	heavy
LocateBinaryLargeObject	Database BLOB Retrieval Time (ms) Time taken to locate the Blob.	heavy
SessionPersistence	Time to Persist Session Data (ms) Time taken to persist the session.	heavy
DeleteArtifact	SAML Artifact Deletion Time (ms) Time taken to delete the artifact	heavy
DeleteProfileState	ProfileState Data Deletion Time (ms) Time taken to delete the Profile State.	heavy
DeleteActiveIdPFederation	Active IdP Federation Deletion Time (ms) Time taken to delete the active IdP federation.	heavy
DeleteActiveSPFederation	Active SP Federation Deletion Time (ms) Time taken to delete the active SP federation.	heavy
DeleteProviderFederation	Provider Federation Deletion Time (ms) Time taken to delete the provider federation.	heavy
ProviderFederationPersistence	Time to Persist a Provider Federation(ms) Time taken to persist the provider federation.	heavy

7.1.2.4.2 Protocol Profiles

Table 7–4 shows the protocol profile metrics collected by phase event sensors for requests and responses:

Note: In the table, the **Label and Description** refers to the short label attached to the metric in Fusion Middleware Control, followed by a description of the metric.

Table 7–4 Protocol Profile Events

Name	Label and Description	Weight
LocalAuthn	Local User Authentication Time (ms) Time taken by the user to get authenticated locally at IdP/SP.	normal
AuthnRequestProcessing	AuthnRequest Processing time at the IdP (ms) Time taken to process AuthnRequest at IdP.	heavy
AuthnResponseProcessing	AuthnResponse Processing Time at SP (ms) Time taken to process AuthnResponse at SP.	normal
ArtifactProcessing	SAML Artifact Processing Time (ms) Time taken to process Artifact.	heavy
Logout	Global Logout Time (ms) Time taken for global logout.	heavy
RequestProcessing	Incoming Request Processing Time (ms) Time taken by ApplicationController to process request.	normal
EventProcessing	Event Processing Time (ms) Time taken by ActionStateMachine to process event.	heavy

7.1.2.4.3 Security Processing

Table 7–5 shows the metrics collected during security processing by phase event sensors:

Note: In the table, the **Label and Description** refers to the short label attached to the metric in Fusion Middleware Control, followed by a description of the metric.

Table 7–5 Security Processing for Phase Events

Name	Label and Description	Weight
XMLSigner	XML Message Signing Time (ms) Time taken by XMLSigner to sign message.	heavy
XMLSignatureVerifier	XML Message Signature Verification Time (ms) Time taken by XMLSignatureVerifier to verify the message signature.	heavy
QueryStringSigner	URL Query String Signing Time (ms) Time taken to sign the Query string.	heavy
QueryStringSignatureVerifier	URL Query String Signature Verification Time (ms) Time taken to verify the signature for Query string.	heavy
XMLEncryptionService	XML Message Encryption Time (ms) Time taken to encrypt the message.	heavy

Table 7–5 (Cont.) Security Processing for Phase Events

Name	Label and Description	Weight
XMLDecryptionService	XML Message Decryption Time (ms) Time taken to decrypt the message.	heavy
SerializeMessage	XML Message Marshalling Time (ms) Time taken by LibertyProtocolMarshaller to serialize the message.	heavy
DeSerializeMessage	XML Message Unmarshalling Time (ms) Time taken by the LibertyProtocolMarshaller to deserialize the message.	heavy

7.2 Availability

Oracle Identity Federation is a Java component whose availability is tracked through Fusion Middleware Control.

For details, see Getting Started Using Oracle Enterprise Manager Fusion Middleware Control in the *Oracle Fusion Middleware Administrator's Guide*.

7.3 Logging

This section describes logging for Oracle Identity Federation:

- About Oracle Identity Federation Logging
- Viewing Oracle Identity Federation Log Messages
- Configuring Oracle Identity Federation Logs
- Common Log Messages

For more information about logging in Oracle Fusion Middleware, see Managing Log Files and Diagnostic Data in the *Oracle Fusion Middleware Administrator's Guide*.

7.3.1 About Oracle Identity Federation Logging

This section provides a basic overview of logging for Oracle Identity Federation. Topics include:

- Types of Logs
- Log Levels
- Message IDs
- Tools for Log Configuration

7.3.1.1 Types of Logs

Oracle Identity Federation provides two types of logs:

- Persistent Logs - These logs persist across component restarts.
- Runtime Logs - These logs are created automatically by the server at runtime and become active when a specific feature is activated.

The persistent log files include:

- `servername-diagnostic.log` - Contains general application log messages, debug messages, and error messages. This log is also referred to as the federation log.

- Other log files that may contain logging messages pertaining to Oracle Identity Federation are *servername*`.log` and *servername*`.out`.

7.3.1.2 Log Levels

Table 7–6 shows the log levels of Oracle Identity Federation log messages:

Table 7–6 Oracle Identity Federation Log levels

Log Level	description
INTERNAL ERROR	Events that represent unrecoverable errors.
ERROR	Events that represent recoverable and unrecoverable errors.
WARNING	Events that represent failures in processing external and implicit Oracle Identity Federation server actions.
NOTIFICATION	High Level Oracle Identity Federation operational events describing a flow.
TRACE	Events with detailed processing flows and state information.

7.3.1.3 Message IDs

Oracle Identity Federation log messages fall into these categories:

Table 7–7 Oracle Identity Federation Message Categories

Message ID Range	Message Category
FED-10000 to FED-10099	Compliance
FED-10100 to FED-10999	Configuration
FED-11000 to FED-11699	Data
FED-11700 to FED-11999	Network
FED-12000 to FED-12999	Other
FED-13000 to FED-14999	Programmatic
FED-15000 to FED-17999	Requests and Responses
FED-18000 to FED-19999	Security
FED-20000 to FED-20099	Threads

7.3.1.4 Tools for Log Configuration

Oracle Identity Federation provides two tools for log configuration and management:

- Fusion Middleware Control for GUI-based configuration
- `wlst` for command-line configuration

7.3.2 Viewing Oracle Identity Federation Log Messages

Log in to Fusion Middleware Control and navigate to Oracle Identity Federation instance. In the Oracle Identity Federation drop down menu, select **Logs**, then **View Log Messages**.

7.3.2.1 Select Messages to View

Take these steps to select messages to view:

1. From the Oracle Identity Federation menu, select **Logs**, then **View Log Messages**. The Log Messages page appears.
2. Select the date range for the logs you want to view. You can select most recent, by minutes, hours or days. Alternatively, you can select a time interval and specify the date and time to start and end.
3. Select the message types you want to view.
4. Specify any additional conditions (such as display only messages that contain some string).
5. To perform a specific search, choose **Add Fields** and add fields on which to search. For each field, select a criterion from the list, then enter text into the box. Choose the red X to delete a field. Choose **Add Fields** to add additional fields. When you have finished adding criteria, click **Search**.

7.3.2.2 Specify View Options

In addition to specifying messages to view, several other viewing options are available:

- Use the Broaden Target Scope list to view messages for the domain.
- Choose **Export Messages to File** to export the log messages to a file as XML, text, or comma-separated list.
- Click **Target Log Files** to view information about individual log files.
- You can indicate when to refresh the view. Select Manual Refresh, 30-Second Refresh, or One Minute Refresh from the list on the upper right.
- Use the View list to change the columns listed or to reorder columns.
- Use the Show list to change the grouping of messages.
- Collapse the **Search** label to view only the list of log messages.
- To view the contents of a log file, double-click the file name in the Log File column. The View Log File: filename page is displayed. You can use the up and down arrows in the Time, Message Type, and Message ID columns to reorder the records in the file.

7.3.3 Configuring Oracle Identity Federation Logs

Use these pages to view and configure Oracle Identity Federation server logs.

Take these steps to navigate to the log configuration page:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. In the Oracle Identity Federation drop-down menu, select **Logs**, then **Log Configuration**.

Topics include:

- Configure Oracle Identity Federation Log Levels
- Configure Oracle Identity Federation Log Files

7.3.3.1 Configure Oracle Identity Federation Log Levels

Use this page to:

- view and update logging levels for Oracle Identity Federation loggers.

- create a new persistent logger.

Each logger logs messages for a specific server function; for example, the EJB deployment logger logs messages for an EJB module.

View or Update Logger Level

Use the View drop-down to select the logger.

Fields include:

- Logger Name - This is the name of the logger.
- Oracle Diagnostic Logging Level - This is the logging level. Use the level drop-down to change the log level.
- Log File - This is the name of the log file. Click on a log file name to view and update the properties of the log file.

Specify a Logger

This portion of the page appears when you select "Loggers with Persistent Log Level" in the View drop-down.

Supply the following information to create a persistent logger:

- Name - Enter a name for the new logger.
- Oracle Diagnostic Logging Level - This is the logging level. Use the level drop-down to select a log level.

Buttons on the page perform the following functions:

- **Apply** - Save the logger configuration updates or generate the new logger information.
- **Revert** - Discard the configuration updates.

7.3.3.2 Configure Oracle Identity Federation Log Files

For information on configuring log files, see Configuring Settings for Log Files in the *Oracle Fusion Middleware Administrator's Guide*.

7.3.4 Common Log Messages

This section explains some common messages you may encounter in the Oracle Identity Federation logs.

7.3.4.1 thread interrupt Messages

You may see a message like the following in the managed server log file:

```
oracle.security.fed.jvt.discovery.model.session.RDBMSSessionDiscoveryProvider
run
WARNING: InterruptedException: thread interrupt occurred during sleep()
java.lang.InterruptedException: sleep interrupted
```

These messages are only notifications indicating that the RDBMS sleeping threads have been killed as a result of a configuration reload; new threads were created to replace these threads. No action is required.

7.4 Auditing

Oracle Identity Federation uses the Fusion Middleware Audit Framework for auditing.

This section explains what events are audited, and how to configure auditing for Oracle Identity Federation. It contains these sections:

- About Auditing in Oracle Identity Federation
- Configuring Auditing for Oracle Identity Federation
- Viewing Audit Data

> **See Also:** Configuring and Managing Auditing in the *Oracle Fusion Middleware Application Security Guide* for details about audit configuration.

7.4.1 About Auditing in Oracle Identity Federation

This section lists the events that you can audit in different categories, and explains audit levels.

7.4.1.1 Categories of Audit Events

There are four categories of audit events for Oracle Identity Federation:

- User Session Management
- Protocol Flow
- Server Configuration
- Security

The events for each category are described in these subsections.

7.4.1.1.1 User Session Management Events

Session management events and the attributes of each event are as follows:

- CreateUserSession – Creation of a user session after a successful login
  - SessionID
  - AuthenticationMechanism
  - UserID
- DeleteUserSession – Deletion of a user session after logout
  - SessionID
  - AuthenticationMechanism
  - UserID
- CreateUserFederation – Creation of a user federation between two remote servers
  - FederationID
  - FederationType (SP/IdP/Affiliation)
  - UserID
  - RemoteProviderID
  - ProtocolVersion

  - NameIDFormat
  - NameIDQualifier
  - NameIDValue
- UpdateUserFederation - Updating the user federation between two remote servers
  - FederationID
  - FederationType (SP/IdP)
  - UserID
  - RemoteProviderID
  - ProtocolVersion
  - NameIDFormat
  - NameIDQualifier
  - NameIDValue
  - OldNameIDQualifier
  - OldNameIDValue
- DeleteUserFederation – Deletion of a user federation between two remote servers
  - FederationID
  - FederationType (SP/IdP)
  - UserID
  - RemoteProviderID
  - ProtocolVersion
  - NameIDFormat
  - NameIDQualifier
  - NameIDValue
- CreateActiveUserFederation – Creation of an active federation after successful login
  - FederationID
  - FederationType (SP/IdP)
  - SessionID
  - UserID
  - RemoteProviderID
  - ProtocolVersion
- DeleteActiveUserFederation - Deletion of an active federation after logout
  - FederationID
  - FederationType (SP/IdP)
  - SessionID
  - UserID
  - RemoteProviderID

  - ProtocolVersion
- LocalAuthentication – Authentication of a user at OIF
  - AuthenticationMechanism
  - AuthenticationEngineID
  - RemoteIP
  - SessionID
  - UserID
- LocalLogout - Logout of a user at Oracle Identity Federation
  - RemoteIP
  - SessionID
  - UserID

7.4.1.1.2 Protocol Flow Events

Protocol flow events and their attributes are as follows:

- IncomingMessage – Message being received by Oracle Identity Federation
  - RemoteIP
  - Binding (for example, SOAP/GET/POST/Artifact/…)
  - ProtocolVersion (for example, SAML2/Libv11/…)
  - RemoteProviderID
  - Role (for example, Service Provider/Identity Provider/Attribute Authority/…)
  - IncomingMessageString (CLOB)
  - MessageType (for example, SSOLoginRequest/SSOLoginResponse/SSOLogoutRequest/…)
- OutgoingMessage - Message being sent by Oracle Identity Federation (Success only)
  - RemoteIP
  - Binding (for example, SOAP/GET/POST/Artifact/…)
  - ProtocolVersion (for example, SAML2/Libv11/…)
  - RemoteProviderID
  - Role (for example, Service Provider/Identity Provider/Attribute Authority/…)
  - OutgoingMessageString (CLOB)
  - MessageType (for example, SSOLoginRequest/SSOLoginResponse/SSOLogoutRequest/…)
- AssertionCreation – Creation of an assertion by Oracle Identity Federation (Success only)
  - RemoteIP
  - ProtocolVersion (for example, SAML2/Libv11/…)
  - AssertionVersion (for example, 2.0)

  - IssueInstant
  - Issuer
  - NameIDQualifier
  - NameIDValue
  - NameIDFormat
  - AssertionID
  - UserID
  - SessionID
  - FederationID
  - RemoteProviderID
- AssertionConsumption - Consumption of an assertion by Oracle Identity Federation (Success only)
  - ProtocolVersion (for example, SAML2/Libv11/…)
  - AssertionVersion (for example, 2.0)
  - IssueInstant
  - Issuer
  - NameIDQualifier
  - NameIDValue
  - NameIDFormat
  - AssertionID
  - UserID
  - SessionID
  - FederationID
  - RemoteProviderID

7.4.1.1.3 Server Configuration Events

Server configuration events and their attributes are as follows:

- CreateConfigProperty – Adding a new configuration property (Success only)
  - PropertyName
  - PropertyType (for example, PropertiesList, PropertiesMap, String, Boolean…)
  - Value
  - PeerProviderID
  - Hierarchy
- ChangeConfigProperty - Changing the value of an existing configuration property(Success only)
  - PropertyName
  - PropertyType (for example, PropertiesList, PropertiesMap, String, Boolean…)
  - OldValue

  - NewValue
  - PeerProviderID
  - Hierarchy
- DeleteConfigProperty - Deleting a configuration property (Success only)
  - PropertyName
  - PropertyType (for example, PropertiesList, PropertiesMap, String, Boolean…)
  - OldValue
  - PeerProviderID
  - Hierarchy
- CreatePeerProvider – Adding a new provider to the list of trusted providers (Success only)
  - PeerProviderID
  - ProviderType (for example, sp, idp, sp idp,…)
  - ProtocolVersion
  - Description
- UpdatePeerProvider - Updating the information on an existing provider in the list of trusted providers (Success only)
- PeerProviderID
  - PeerProviderID
  - ProviderType (for example, sp, idp, sp idp,…)
  - ProtocolVersion
  - Description
- DeletePeerProvider - Deleting a provider from the list of trusted providers (Success only
  - PeerProviderID
  - ProviderType (for example, sp, idp, sp idp,…)
  - ProtocolVersion
  - Description
- LoadMetadata – Loading of metadata (Success only)
  - Metadata
  - Description
- SetDataStoreType – Changing the type of a data store (Success only)
  - DataStoreName
  - OldValue
  - NewDataStoreType
- ChangeDataStore – Setting of the federation data store (Success only)
  - DataStoreBefore
  - DataStoreAfter

- ChangeFederation – Changing of the trusted providers (Success only)
  - COTBefore
  - COTAfter
- ChangeServerProperty – Changing of a server configuration property (Success only)
  - ServerConfigBefore
  - ServerConfigAfter

7.4.1.1.4 Security Events

Security events and their attributes are as follows:

- CreateSignature – Creation of a digital signature by Oracle Identity Federation
  - Type (XML, String)
- VerifySignature – Verification of a digital signature by Oracle Identity Federation
  - Type (XML, String)
- EncryptData – Encryption of data by Oracle Identity Federation
  - Type (XML, String)
- DecryptData – Decryption of data by Oracle Identity Federation
  - Type (XML, String)

7.4.1.1.5 Attributes Shared by All Events In addition there are attributes shared for all events:

- timestamp - the timestamp of when the audit event occurred
- initiator - the initiator of the audit event (for some events this attribute may be empty)
- ECID - the execution context ID

7.4.1.2 Audit Levels

Fusion Middleware Audit Framework supports the following audit levels:

- None
- Low
- Medium
- Custom

The following audit events get audited at the Low and Medium audit levels:

> **Note:** FAILURESONLY denotes that the event will only get audited in case of failure.

Events Audited at Low level

- ServerConfiguration
  - CreateConfigProperty
  - ChangeConfigProperty

  - DeleteConfigProperty
  - CreatePeerProvider
  - UpdatePeerProvider
  - DeletePeerProvider
  - LoadMetadata
  - SetDataStoreType
  - ChangeDataStore
  - ChangeCOT
  - ChangeServerProperty

Events Audited at Medium level

- ServerConfiguration
  - CreateConfigProperty
  - ChangeConfigProperty
  - DeleteConfigProperty
  - CreatePeerProvider
  - UpdatePeerProvider
  - DeletePeerProvider
  - LoadMetadata
  - SetDataStoreType
  - ChangeDataStore
  - ChangeCOT
  - ChangeServerProperty
- UserSession.FAILUREONLY
  - CreateUserSession.FAILUREONLY
  - DeleteUserSession.FAILUREONLY
  - CreateUserFederation.FAILUREONLY
  - UpdateUserFederation.FAILUREONLY
  - DeleteUserFederation.FAILUREONLY
  - CreateActiveUserFederation.FAILUREONLY
  - DeleteActiveUserFederation.FAILUREONLY
  - LocalAuthentication.FAILUREONLY
  - LocalLogout.FAILUREONLY
- ProtocolFlow.FAILUREONLY
  - IncomingMessage.FAILUREONLY
  - OutgoingMessage.FAILUREONLY
  - AssertionCreation.FAILUREONLY
  - AssertionConsumption.FAILUREONLY

- Security.FAILUREONLY
  - CreateSignature.FAILUREONLY
  - VerifySignature.FAILUREONLY
  - EncryptData.FAILUREONLY
  - DecryptData.FAILUREONLY

Events Audited at Custom Level

The Custom audit level allows you to select only the events you wish to audit.

> **See Also:** Configuring and Managing Auditing in the *Oracle Fusion Middleware Application Security Guide*.

7.4.2 Configuring Auditing for Oracle Identity Federation

You can use Oracle Enterprise Manager Fusion Middleware Control or `WLST` command-line interface to configure auditing.

Take these steps to get started with configuring auditing with Fusion Middleware Control:

1. Log in to Fusion Middleware Control and navigate to the Identity Management domain.
2. In the Weblogic Domain drop down menu, select **Security**, then **Audit Policy**.
3. Select the Oracle Identity Federation component.
4. In the Audit Level menu, select the desired audit level.

   You can view the audit policies that will be enforced in different categories by expanding the + check-box for the component.

   > **Note:** If selected level is Custom, refer to Section 7.4.2.1, "Configuring Auditing at the Custom Level".

5. Optionally, in the Users text box, you can add users who will always be audited for all events, regardless of audit level.
6. Click **Apply**.

7.4.2.1 Configuring Auditing at the Custom Level

Take these steps if you are configuring audit policies and wish to use the Custom audit level:

1. In the Audit Level menu, select **Custom** as the audit level.
2. Select the events to audit in the table of events:
   - Click the **+** sign next to the component name to get the list of audit event categories.
   - Click the **+** sign next to the category name to get the list of events.
   - Click the **+** sign next to the event name to get Success/Failure audit options.
3. Check the **Enable Audit** box next to the events or categories desired to audit. (for example, checking the box next to Security will audit all security events. Checking

the box next to `CreateUserSession Failure` event will audit all `CreateUserSession` failure events.

4. Optionally, you can add filters for fine-grained auditing.

   Click the pencil icon to the right of the event or category name. Add the desired filter conditions.

5. Click **OK** when finished.

> **See Also:** Configuring and Managing Auditing in the *Oracle Fusion Middleware Application Security Guide* for details about audit policy configuration.

7.4.3 Viewing Audit Data

Your audit data may reside in files (also known as bus-stop files), or it may reside in a database audit store.

If the audit data resides in a bus-stop file, you can query the file directly at this location:

```
<domain_home>/servers/<server_name>/logs/auditlogs/OIF/audit.log
```

If the audit data resides in a database, you can use a tool like Oracle Business Intelligence Publisher to view audit reports.

> **See Also:** Using Audit Analysis and Reporting in the *Oracle Fusion Middleware Application Security Guide*.

8 Security

This chapter describes Oracle Identity Federation security topics, including:

- Configuring SSL for Oracle Identity Federation
- Managing Signing and Encryption Wallets
- Setting up JCE Policy Files for Oracle WebLogic Server

8.1 Configuring SSL for Oracle Identity Federation

About Keystore Usage in the Server

Oracle Identity Federation only supports configuring one password for signing and encryption keystores, and uses that password to open both the keystore and the private key. This means that if a keystore is configured with different store password and key password, an error will occur when Oracle Identity Federation tries to access the private key.

To avoid this error, ensure that the private key password for the configured key alias is the same as the keystore password.

Note: In Oracle Identity Federation 11g Release 1 (11.1.1), if you change the key password to match the keystore password, you must remove the old keystore/wallet from the configuration.

Note: Keystores, trusted certificates and certificates for Oracle Identity Federation are managed the same way as they are for any other Oracle Fusion Middleware component. For details, see the *Oracle Fusion Middleware Administrator's Guide*.

This section contains these topics:

- Configuring Oracle Identity Federation as an SSL Server
- Configuring Oracle Identity Federation as an SSL Client
- Configuring SSL in HA Mode

See Also: Section 6.4.3 for information about configuring Oracle Identity Federation HA in SSL mode.

8.1.1 Configuring Oracle Identity Federation as an SSL Server

This section explains how to configure the SSL port for Oracle WebLogic Server, and how to configure Oracle Identity Federation to use SSL.

8.1.1.1 Setting up SSL on Oracle WebLogic Server

Take these steps to configure the SSL port and keystore for the Oracle WebLogic Server for which you are setting up SSL:

1. Log in to the Oracle WebLogic Server administration console and navigate to **Environment**, then **Servers**.
2. Select the server for which you want to set up SSL.
3. Check **SSL Listen Port Enabled** and enter an SSL listening port number (for example. 443).

   We will subsequently refer to this port as `$SSL_PORT`.
4. Click **Save**.
5. Go to the Keystores tab, and click **Lock & Edit**.
6. In Keystores, select an option that includes Custom Identity.
7. In the Identity section, fill in properties as follows:
   - Custom Identity Keystore: *`location of keystore containing the SSL private key and certificate`*
   - Custom Identity Keystore type: `jks`
   - Custom Identity Keystore Passphrase: *`storepassword`*
8. Click **Save**.
9. Go to the SSL tab.
10. In the Identity section, fill in properties as follows:
    - Private Key Alias: *`keyalias`*
    - Private Key Passphrase: *`keypassword`*
11. Click **Save**, then click **Activate Changes**.
12. Restart the server.
13. To verify that SSL was set up correctly, go to `https://`*`$HOSTNAME:$SSL_PORT`*; a certificate should be presented. View the certificate; the subject should match the `cn` entered when creating the certificate.

Notes:

- The CN of the SSL server certificate must be the fully qualified hostname, for example eaevma1302.de.mycorp.com, not eaevma1302.
- For complete information on how to set up SSL on Oracle Weblogic Server, refer to Configuring SSL in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

If you wish to configure Oracle WebLogic Server to require a client SSL certificate, take the following steps:

1. Log in to the Oracle WebLogic Server administration console and navigate to Environment, then Servers.
2. Select the server for which you want to set up SSL.
3. Go to the **SSL** tab, then **Advanced**.
4. For the property "Two Way Client Cert Behavior", select "Client Certs Requested and Enforced".
5. Click **Save**.
6. Go to the Keystores tab.
7. In Keystores, select an option that has the type of Trust Keystore type you wish to configure, and populate the fields in the Trust section.
8. Click **Save**, and click **Activate Changes**.
9. Restart the server.

You will need to import the CA that issued the client certificate into the Trust Keystore you specified in the Oracle WebLogic Configuration.   If it is a Java Keystore, you can use the following command:

```
keytool -import -alias aliasfortrustedca -trustcacerts -file trustedcafile.pem
-keystore keystorelocation -storepass truststorepassword
```

8.1.1.2 Configuring Oracle Identity Federation

Once you have enabled an SSL listening port and uploaded the server and trusted certificates to the respective keystores, you will need to configure Oracle Identity Federation to use SSL.

Follow these steps:

1. Log in to Fusion Middleware Control and locate the Oracle Identity Federation instance.
2. Navigate to Server Properties.
3. Update the port (and SOAP port, if necessary) to reflect the SSL port configured in the Oracle Weblogic Server administration console.
4. Check the **SSL Enabled** checkbox.
5. To force the use of SSL if a request is received at a non-SSL port, check the **Force SSL** box. Leave unchecked otherwise.
6. To force client authentication, check the **Require Client Certificate** box. Leave unchecked otherwise.
7. Click **Apply**.

You must re-generate and re-distribute metadata to peer providers after enabling SSL.

Notes:

- Changing the port (and SOAP port) modifies the server's metadata to reflect the correct service URLs.
- The metadata at the peer providers' sites must be updated with the new version.

8.1.2 Configuring Oracle Identity Federation as an SSL Client

There are two ways to configure Oracle Identity Federation as an SSL client to connect to remote SSL servers:

- Set up Oracle Identity Federation to use the Oracle WebLogic Server keystores as its identity and trust repositories. This approach is described in Section 8.1.2.1, "Configuring Oracle WebLogic Server" and Section 8.1.2.2, "Configuring Keystore Passwords in Oracle Identity Federation".
- Set up Oracle Identity Federation to use its own identity and trust keystores. This approach is described in Section 8.1.2.3, "Alternative Way to Configure Oracle Identity Federation as SSL Client".

Topics in this section include:

- Configuring Oracle WebLogic Server
- Configuring Keystore Passwords in Oracle Identity Federation
- Alternative Way to Configure Oracle Identity Federation as SSL Client
- Connecting to an LDAP Server over SSL
- Ensuring that Fusion Middleware Control can Manage an Oracle Identity Federation Target

8.1.2.1 Configuring Oracle WebLogic Server

Some SSL servers might require authentication of the client performed during the SSL handshake. This operation is typically done by having the SSL client present an SSL Client certificate to the SSL server.

This section describes how to configure Oracle WebLogic Server and Oracle Identity Federation to present a Client SSL certificate when it is requested by an SSL server. This requires:

- setting up trust for the CA that issued the SSL server certificates
- obtaining a certificate for the Oracle Identity Federation SSL client.

Take these steps to achieve this:

1. Log in to the Oracle WebLogic Server administration console and navigate to **Environment**, then **Servers**.
2. Select the server for which you want to set up SSL.
3. Go to the **Keystores** tab, and click **Lock & Edit**.
4. In **Keystores**, select an option that includes Custom Identity and the Trust Keystore type you wish to configure.
5. In the Identity section, fill in properties as follows:
   - Custom Identity Keystore: *`location of keystore with SSL private key and certificate`*
   - Custom Identity Keystore type: *`identity keystore type`*
   - Custom Identity Keystore Passphrase: *`storepassword`*
6. In the **Trust** section, fill in the properties with the Trust Keystore information.
7. Click **Save**, then click **Activate Changes**.
8. Restart the server.

8.1.2.2 Configuring Keystore Passwords in Oracle Identity Federation

If Oracle Identity Federation needs to connect to a remote provider and provide an SSL client certificate, you must configure the identity and trust keystore passwords in Oracle Identity Federation setup, not in Oracle WebLogic Server. Follow these steps:

1. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Server Properties**.
3. In the Outbound Connections section under SSL Settings, enter the values of these two properties:
   - WebLogic Server Identity Keystore Password - the password of the identity keystore you entered in the Oracle WebLogic Server configuration.
   - WebLogic Server Trust Keystore Password - the password of the trust keystore you entered in the Oracle WebLogic Server configuration. If this property is left empty, the trust keystore will be opened without a password.

8.1.2.3 Alternative Way to Configure Oracle Identity Federation as SSL Client

If you do not wish to enter identity and trust keystore information in the Oracle WebLogic Server configuration, there is an alternate way to configure Oracle Identity Federation as an SSL Client when connecting to remote SSL servers.

With this approach, you will need to use the Oracle Identity Federation `WLST` commands or MBeans to set certain configuration properties. You will also need to enter the keystore passwords in the credential store.

8.1.2.3.1 Setting properties in Oracle Identity Federation configuration

You will need to set these five "serverconfig" properties to the following values:

- usewlssslconfig - `false`
- clientsslkeystoreloc - the path and filename of the identity keystore. The path can be absolute or relative to the domain home.
- clientsslkeystoretype – the identity keystore type. If no type is specified, the type is assumed to be JKS.
- clientssltruststoreloc – the path and filename of the trust keystore. The path can be absolute or relative to the domain home.
- clientssltruststoretype – the trust keystore type. If no type is specified, the type is assumed to be JKS.

Example: Using the WLST commands

```
setConfigProperty('serverconfig', 'usewlssslconfig', 'false', 'BOOLEAN')
setConfigProperty('serverconfig', 'clientsslkeystoreloc',
   '/usr/local/ssl/keystore', 'STRING')
setConfigProperty('serverconfig', 'clientsslkeystoretype', 'JKS', 'STRING')
setConfigProperty('serverconfig', 'clientssltruststoreloc',
   '/usr/local/ssl/truststore', 'STRING')
setConfigProperty('serverconfig', 'clientssltruststoretype', 'JKS', 'STRING')
```

See Chapter 9, "Oracle Identity Federation Command-Line Tools" for details about `WLST` command usage.

Example: Using the MBeans

In the ConfigMXBean with name "serverconfig", invoke the "putProperty" operation five times with the following arguments:

Property Name	Property Value	Property Type
usewlssslconfig	false	BOOLEAN
clientsslkeystoreloc	/usr/local/ssl/keystore	STRING
clientsslkeystoretype	JKS	STRING
clientsslstruststoreloc	/usr/local/ssl/keystore	STRING
clientssltruststoretype	JKS	STRING

See Appendix A, "Oracle Identity Federation MBeans" for details.

8.1.2.3.2 Entering keystore passwords in the credential store

You will need to store the identity and trust keystore passwords in the credential store. The keys for these passwords in the credential store are:

- clientsslkeystorepwd – the password of the Identity Keystore
- clientssltruststorepwd – the password of the Trust Keystore

Following is an example of how to use `WLST` commands to create and update these passwords in the credential store. This example assumes that Oracle Identity Federation is deployed with application name "OIF"; the password of both the Identity and Trust keystore is denoted as "mypassword".

Create the keystore credentials:

```
createCred(map="OIF", key="clientsslkeystorepwd",
user="UniqueUserNameCredential", password="mypassword", desc="identity keystore
pwd")

createCred(map="OIF", key="clientssltruststorepwd",
user="UniqueUserNameCredential", password="mypassword", desc="trust keystore pwd")
```

Update the keystore credentials:

```
updateCred(map="OIF", key="clientsslkeystorepwd",
user="UniqueUserNameCredential", password="mypassword", desc="identity keystore
pwd")

updateCred(map="OIF", key="clientssltruststorepwd",
user="UniqueUserNameCredential", password="mypassword", desc="trust keystore pwd")
```

See Section 4.5, "Managing Credentials for Oracle Identity Federation" for details.

8.1.2.4 Connecting to an LDAP Server over SSL

When Oracle Identity Federation needs to connect to an LDAP server using SSL, you first need to add the LDAP's CA certificate to the trust keystore in the Oracle WebLogic Server Administration Console; this information is provided on the Server/Keystores configuration screen for the managed server where Oracle Identity Federation is running.

You must also enter the trust keystore password in Oracle Identity Federation configuration (See Section 8.1.2.1, "Configuring Oracle WebLogic Server" and Section 8.1.2.2, "Configuring Keystore Passwords in Oracle Identity Federation").

Notes:

- Oracle Identity Federation does not support client authentication when connecting to LDAP servers.
- Oracle Identity Federation will only use the `WLS` trust keystore when connecting to LDAP servers.

When Searching LDAP Server over SSL

If the user and/or federation data stores are LDAP servers using SSL, and you wish to use the search operations in Fusion Middleware Control (navigate to **Administration**, then **Identities**), you will need to import the LDAP's CA certificate to the JVM's `cacert` keystore.

When performing the search operation, you will see the following error printed in the logs:

```
SEVERE: NamingException: error while interacting with an LDAP server or JNDI
module
javax.naming.NameNotFoundException: remaining name: env/jmx/runtime
```

This is expected and will not affect the search.

8.1.2.5 Ensuring that Fusion Middleware Control can Manage an Oracle Identity Federation Target

After SSL is enabled for the Admin server and the managed server hosting Oracle Identity Federation, you must ensure that Fusion Middleware Control can continue to manage the Oracle Identity Federation server.

Take these steps to enable Fusion Middleware Control to manage an Oracle Identity Federation server target:

1. Locate $INSTANCE_HOME/EMAGENT/EMAGENT/sysman/emd/targets.xml.

   Change the protocol for the 'serviceURL' property to the correct protocol. If you have more than one Oracle Identity Federation target (besides host and oracle_emd), you need to modify the 'serviceURL' for each target.

2. Locate $INSTANCE_HOME/EMAGENT/EMAGENT/sysman/config/emd.properties.

   If necessary, update the protocol for 'REPOSITORY_URL' to the correct protocol. The EM Agent uses this property to connect to Fusion Middleware Control.

3. Stop the EM Agent using the command:

   ```
   $INSTANCE_HOME/bin/opmnctl stopproc ias-component=EMAGNET
   ```

4. Secure the EM Agent using the command:

   ```
   $INSTANCE_HOME/EMAGENT/EMAGENT/bin/emctl secure fmagent -admin_host
   <host> -admin_port <port> -admin_user <username> [-admin_pwd <pwd>]
   ```

5. Restart the EM Agent using the command:

   ```
   $INSTANCE_HOME/bin/opmnctl startproc ias-component=EMAGNET
   ```

8.1.3 Configuring SSL in HA Mode

For details on this topic, see Section 6.4.3.

8.2 Managing Signing and Encryption Wallets

Oracle Identity Federation provides a way to update signing and/or encryption wallets smoothly, without interrupting service.

When you need to replace a signing or encryption wallet and a new one is uploaded, Oracle Identity Federation saves the old wallet. The server then continues to use the old wallet in all transactions until it is removed. However, generated metadata will contain the new wallet information and the old information. This allows time to notify remote providers about the change.

Once new metadata has been created and distributed to all remote providers, the old wallet can be deleted and Oracle Identity Federation will use the newly uploaded wallet for all subsequent transactions.

This section contains these topics:

- Signing and Encryption Passwords
- Replacing a Signing or Encryption Wallet

8.2.1 Signing and Encryption Passwords

As of 11*g* Release 1 (11.1.1) Patch Set 3, the keystore (signing key) password and the encryption key password do not need to be the same. The treatment of passwords is as follows:

- You can configure distinct store password and key password.
- If not configured, the key password is assumed to be the same as the store password.

> **See Also:** Managing Keystores, Wallets, and Certificates in the *Oracle Fusion Middleware Administrator's Guide* for details about keystore management.

8.2.2 Replacing a Signing or Encryption Wallet

Follow these steps when replacing a signing or encryption wallet:

1. Upload the new wallet.
   a. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
   b. Navigate to **Administration**, then **Security and Trust**.
   c. In the **Wallets** tab, click **Update**.
   d. Check the **Update** checkbox for the wallet you want to update.
   e. Select the keystore type, wallet location, password, and alias.
   f. Click **OK**.
2. Generate and distribute new metadata.
   a. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.

   b. Navigate to **Administration**, then **Security and Trust**.
   c. In the **Provider Metadata** tab, under the **Generate Metadata** section, select the provider type and the protocol of the metadata to be generated, and click **Generate**.
   d. Save the generated metadata.
   e. Distribute the generated metadata to all remote peer providers.
3. Delete the old wallet.
   a. Log in to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
   b. Navigate to **Administration**, then **Security and Trust**.
   c. In the **Wallets** tab, click **Update**.
   d. In the wallet that you have updated, click **Delete old Wallet**.

8.3 Setting up JCE Policy Files for Oracle WebLogic Server

By default, Oracle Identity Federation supports low-strength cryptographic key sizes for encryption/decryption operations such as XML encryption.

In order to use strong symmetric encryption algorithms, such as AES-256, you need to modify the JVM to include the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction policy.

Take these steps:

1. Download Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction policy files from this URL:

   ```
   http://www.oracle.com/technetwork/java/javase/downloads/index.html
   ```

2. Unzip the files in all the `$JAVA_HOME/jre/lib/security` directories located under the `$BEA_HOME` folder (to find those directories, look for `US_export_policy.jar` files). For every `$JAVA_HOME/jre/lib/security` directory, overwrite the default low strength `local_policy.jar` and `US_export_policy.jar` files with the ones provided by Oracle.
3. Restart the administration server and the managed server where Oracle Identity Federation is running.

9 Oracle Identity Federation Command-Line Tools

This chapter describes the command-line tools available for Oracle Identity Federation.

- Introduction to Command-Line Tools for Oracle Identity Federation
- Oracle Identity Federation Commands

9.1 Introduction to Command-Line Tools for Oracle Identity Federation

`WLST` is the command-line utility for administering Oracle Fusion Middleware components and applications. It provides another option for administration in addition to Oracle Enterprise Manager Fusion Middleware Control.

> **See Also:** *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference* for detailed background and explanation of the WLST utility.

The `WLST` command-line utility provides a complete range of tools to manage Oracle Identity Federation, including commands for:

- Property Management
- Federation Management
- Custom SP and Authentication Engine Management
- Message Store Maintenance

9.1.1 Setting up the WLST Environment

Execute the following commands to set up the environment so you can run the `WLST` commands:

On Linux

The syntax to set up the environment on Linux systems is:

```
bash

export $DOMAIN_HOME=PATH_TO_DOMAIN_HOME

source $ORACLE_HOME/fed/scripts/setOIFEnv.sh
(replace $ORACLE_HOME with the correct path for your environment.)
```

On Windows

The syntax to set up the environment on Windows systems is:

```
set DOMAIN_HOME=PATH_TO_DOMAIN_HOME

%ORACLE_HOME%/fed/scripts/setOIFEnv.cmd
```

9.1.2 Executing the Commands

Execute the following command to enter the `WLST` script environment for Oracle Identity Federation:

On Windows

```
IDM_ORACLE_HOME\common\bin\wlst.cmd
```

On Linux

```
IDM_ORACLE_HOME/common/bin/wlst.sh
```

To execute a command, use the format:

```
command-name('param1','param2',...)
```

For example:

```
deleteUserFederations(['user1','user2'])
```

Note: when prompted for the connect() URL, enter the managed server port, not the administration server port.

9.2 Oracle Identity Federation Commands

Use the WLST commands listed in Table 9–1 to view and manage the configuration for Oracle Identity Federation.

Table 9–1 WLST Commands for Oracle Identity Federation

Use this command...	To...	Use with WLST...
addConfigListEntryInMap	Add a configuration list entry to a map.	Online
addConfigMapEntryInMap	Add a configuration map entry to a map.	Online
addConfigPropertyListEntry	Add a configuration property list entry.	Online
addConfigPropertyMapEntry	Add a configuration property map entry.	Online
addCustomAuthnEngine	Add a custom authentication engine.	Online
addCustomSPEngine	Add a custom SP engine.	Online
addFederationListEntryInMap	Add a list entry to a map for a specific remote provider's configuration.	Online
addFederationMapEntryInMap	Add a map entry to a map for a specific remote provider's configuration.	Online
addFederationPropertyListEntry	Add a property list entry for a specific remote provider's configuration.	Online
addFederationPropertyMapEntry	Add a map entry for a specific remote provider's configuration.	Online

Table 9–1 (Cont.) WLST Commands for Oracle Identity Federation

Use this command...	To...	Use with WLST...
deleteCustomAuthnEngine	Delete a custom authentication engine.	Online
deleteCustomSPEngine	Delete a custom SP engine.	Online
deleteProviderFederation	Delete the federated identities for a specific provider.	Online
deleteUserFederations	Delete the federated identities for a specific user.	Online
changeMessageStore	Change the message store to memory or RDBMS.	Online
changePeerProviderDescription	Change a peer provider's description.	Online
changeSessionStore	Change the session store to memory or RDBMS.	Online
createConfigPropertyList	Create a configuration property list.	Online
createConfigPropertyListInMap	Create a configuration property list in the map.	Online
createConfigPropertyMap	Create a configuration property map.	Online
createConfigPropertyMapInMap	Create a nested configuration property map in a map.	Online
createFederationPropertyList	Create a property list for a specific remote provider's configuration.	Online
createFederationPropertyListInMap	Create a property list in a map for a specific remote provider's configuration.	Online
createFederationPropertyMap	Create a property map for a specific remote provider's configuration.	Online
createFederationPropertyMapInMap	Create a nested property map in a map for a specific remote provider's configuration.	Online
createPeerProviderEntry	Create a peer provider entry.	Online
getConfigListValueInMap	Retrieve a configuration list value from a map.	Online
getConfigMapEntryInMap	Retrieve a configuration map value from a map.	Online
getConfigProperty	Retrieve a configuration property entry.	Online
getConfigPropertyList	Retrieve a configuration property list.	Online
getConfigPropertyMapEntry	Retrieve a configuration property map entry.	Online
getFederationListValueInMap	Retrieve a property list value from a map for a specific remote provider's configuration.	Online
getFederationMapEntryInMap	Retrieve a property map value from a map for a specific remote provider's configuration.	Online
getFederationProperty	Retrieve a property value for a specific remote provider's configuration.	Online
getFederationPropertyList	Retrieve a property list for a specific remote provider's configuration.	Online

Table 9–1 (Cont.) WLST Commands for Oracle Identity Federation

Use this command...	To...	Use with WLST...
extractproviderprops	Export all provider configuration properties to a text file.	Script
setproviderprops	Set a provider's properties based on an input text file.	Script
getFederationPropertyMapEntry	Retrieve a property map entry for a specific remote provider's configuration.	Online
listCustomAuthnEngines	Display the list of custom authentication engines.	Online
listCustomSPEngines	Display the list of custom SP engines.	Online
loadMetadata	Load metadata from a file.	Online
oifStatus	Display the status of an Oracle Identity Federation server.	Online
removeConfigListInMap	Delete a configuration list in a map.	Online
removeConfigMapEntryInMap	Delete a configuration map entry in a map.	Online
removeConfigMapInMap	Delete a nested configuration map.	Online
removeConfigProperty	Delete a configuration property.	Online
removeConfigPropertyList	Delete a property list.	Online
removeConfigPropertyMap	Delete a property map.	Online
removeConfigPropertyMapEntry	Delete an entry in the property map.	Online
removeFederationListInMap	Delete a list from a map for a specific remote provider's configuration.	Online
removeFederationMapInMap	Delete a nested map from a map for a specific remote provider's configuration.	Online
removeFederationMapEntryInMap	Delete a nested map property entry from a map for a specific remote provider's configuration.	Online
removeFederationProperty	Delete a property for a specific remote provider's configuration.	Online
removeFederationPropertyList	Delete a property list for a specific remote provider's configuration.	Online
removeFederationPropertyMap	Delete a property map.	Online
removeFederationPropertyMapEntry	Delete a property from a map for a specific remote provider's configuration.	Online
removePeerProviderEntry	Delete a peer provider entry.	Online
setConfigProperty	Set a configuration property.	Online
setCustomAuthnEngine	Define a custom authentication engine.	Online
setCustomSPEngine	Define a custom SP engine.	Online
setFederationProperty	Set a property for a specific remote provider's configuration.	Online

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

9.2.1 addConfigListEntryInMap

Online command that adds a property value to a nested list inside a map.

9.2.1.1 Description

This command adds a property value to a nested list inside a map in config.xml.

9.2.1.2 Syntax

```
addConfigListEntryInMap('configName', 'mapname', 'listName', 'value', 'type')
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
mapname	Specifies the name of the property to map to be changed in config.xml.
listname	Specifies the name of the list.
value	Specifies the property value.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.1.3 Example

The following command adds `valueA` to a map list in server configuration:

```
wls:/mydomain/serverConfig>
addConfigListEntryInMap('serverconfig','mymap','mylistA','valueA','string')
```

9.2.2 addConfigMapEntryInMap

Online command that adds a nested map property entry in a map.

9.2.2.1 Description

This command that adds a property name/value pair to a map nested inside a map in config.xml.

9.2.2.2 Syntax

```
addConfigMapEntryInMap('configName', 'mapname', 'nestedMapName', 'propName',
'value', 'type')
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
mapname	Specifies the name of the property map to be changed in config.xml.
nestedmapname	name of the nested property map to be changed.
propname	Specifies the name of the list.
value	Specifies the property value.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.2.3 Example

The following command adds a boolean name/value pair to `nestedmapB` inside the map `mymap`.

```
wls:/mydomain/serverConfig>
addConfigMapEntryInMap('serverconfig','mymap','nestedmapB','myvarB','true',
'boolean')
```

9.2.3 addConfigPropertyListEntry

Online command that adds a list property entry to config.xml.

9.2.3.1 Description

This command adds a property value to a list in config.xml.

9.2.3.2 Syntax

```
addConfigPropertyListEntry('configName', 'listName', 'value', 'type')
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
listname	Specifies the name of the property list to be updated in config.xml.
value	Specifies the new property list value. The entered value is appended to the list.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.3.3 Example

The following command adds a string value to `mylistA`.

```
wls:/mydomain/serverConfig>
addConfigPropertyListEntry('serverconfig','mylistA','valueA','string')
```

9.2.4 addConfigPropertyMapEntry

Online command that adds a property name/value entry in a map in config.xml.

9.2.4.1 Description

This command adds a property name/value entry in a map in config.xml.

9.2.4.2 Syntax

```
addConfigPropertyMapEntry('configName', 'mapName', 'propName','value', 'type')
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
mapname	Specifies the name of the property map in config.xml.
propname	Specifies the name of the property map.
value	Specifies the property map value to be added.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.4.3 Example

The following command adds `valueA` of string type to a map.

```
wls:/mydomain/serverConfig>
addConfigPropertyMapEntry('serverconfig','mymapA','myvarA','valueA','string')
```

9.2.5 addCustomAuthnEngine

Online command that adds a custom authentication integration engine.

9.2.5.1 Description

This command adds a custom authentication integration engine to config.xml.

9.2.5.2 Syntax

```
addCustomAuthnEngine('name' 'enabled' 'webContext' 'authnRelativePath'
'logoutRelativePath' 'logoutEnabled')
```

Argument	Definition
name	Specifies the name of the custom engine.
enabled	This flag specifies whether the engine is enabled (true) or not (false).
webContext	Specifies the web context for the engine.
authnRelativePath	Specifies the authentication relative path URL for the engine.
logoutRelativePath	Specifies the logout relative path URL for the engine.
logoutEnabled	This flag is set true to enable logout for the engine, else false.

9.2.5.3 Example

The following command defines an engine named `test` and enables it.

```
wls:/mydomain/serverConfig> addCustomAuthnEngine('test','true')
```

9.2.6 addCustomSPEngine

Online command that adds a custom service provider (SP) engine.

9.2.6.1 Description

This command adds a custom SP integration engine to config.xml.

9.2.6.2 Syntax

```
addCustomSPEngine('name' 'enabled' 'authnMech' 'webContext' 'authnRelativePath'
'logoutRelativePath' 'logoutEnabled')
```

Argument	Definition
name	Specifies the name of the custom engine.
enabled	This flag specifies whether the engine is enabled (true) or not (false).
authnMech	Specifies the authentication mechanism for the engine.
webContext	Specifies the web context for the engine.
authnRelativePath	Specifies the authentication relative path URL for the engine.
logoutRelativePath	Specifies the logout relative path URL for the engine.

Argument	Definition
logoutEnabled	This flag is set true to enable logout for the engine, else false.

9.2.6.3 Example

The following command adds an engine and gives it a disabled status.

```
addCustomSPEngine('new
engine','false','oracle:fed:authentication:unspecified','webcontext')
```

9.2.7 addFederationListEntryInMap

Online command that adds a list property entry in a map.

9.2.7.1 Description

This command adds a property value to a nested list inside a map in cot.xml.

9.2.7.2 Syntax

```
addFederationListEntryInMap('providerID', 'mapname', 'listName', 'value', 'type')
```

Argument	Definition
providerID	Specifies the provider ID.
mapname	Specifies the name of the property map to be changed in cot.xml.
listname	Specifies the name of the property list to be added to the map.
value	Specifies the property list value to be added. The entered value is appended to the list.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.7.3 Example

The following command adds a boolean property list to `mymap`.

```
wls:/mydomain/serverConfig>
addFederationListEntryInMap('providerB','mymap','mylistB','true','boolean')
```

9.2.8 addFederationMapEntryInMap

Online command that adds a nested map property entry in a map.

9.2.8.1 Description

This command adds a property name/value pair to a map nested inside a map in cot.xml.

9.2.8.2 Syntax

```
addFederationMapEntryInMap('providerID', 'mapname', 'nestedMapName', 'propName',
'value', 'type')
```

Argument	Definition
providerID	Specifies the provider ID.
mapname	Specifies the name of the property map to be changed in cot.xml.

Argument	Definition
nestedMapName	Specifies the name of the nested property map to be changed.
propName	Specifies the name of the property to be updated in the map.
value	Specifies the property value to be added. The entered value is appended to the list.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.8.3 Example

The following command adds a value of type string to the `myvarA` property in a nested map.

```
wls:/mydomain/serverConfig>
addFederationMapEntryInMap('providerA','mymap','nestedmapA','myvarA','valueA',
'string')
```

9.2.9 addFederationPropertyListEntry

Online command that adds a list property entry.

9.2.9.1 Description

This command adds a property value to a list in cot.xml.

9.2.9.2 Syntax

```
addFederationPropertyListEntry('providerID', 'listName', 'value', 'type')
```

Argument	Definition
providerID	Specifies the provider ID.
listname	Specifies the name of the property list to be updated.
value	Specifies the property list value to be added. The entered value is appended to the list.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.9.3 Example

The following command adds a value in string format to a specified property list.

```
wls:/mydomain/serverConfig>
addFederationPropertyListEntry('providerA','mylistA','valueA','string')
```

9.2.10 addFederationPropertyMapEntry

Online command that a property name/value entry in a map.

9.2.10.1 Description

This command adds a property name/value pair to a map in cot.xml.

9.2.10.2 Syntax

```
addFederationPropertyMapEntry('providerID', 'mapName', 'propName','value', 'type')
```

Argument	Definition
providerID	Specifies the provider ID.
mapname	Specifies the name of the property map to be changed in cot.xml.
propName	Specifies the name of the property to be added in the map.
value	Specifies the property value to be added.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.10.3 Example

The following command adds boolean property `myvarB` to a map.

```
wls:/mydomain/serverConfig>
addFederationPropertyMapEntry('providerA','mymapB','myvarB','true','boolean')
```

9.2.11 deleteCustomAuthnEngine

Online command that deletes a custom authentication integration engine from the configuration.

9.2.11.1 Description

This command deletes a custom authentication integration engine in `config.xml`. You must provide the engine ID for an existing custom authentication engine in `config.xml`.

9.2.11.2 Syntax

```
deleteCustomAuthnEngine('engineID')
```

Argument	Definition
engineID	Specifies the engine ID of an existing engine to be deleted.

9.2.11.3 Example

The following command deletes the authentication engine with ID `id1234`.

```
wls:/mydomain/serverConfig> deleteCustomAuthnEngine('id1234')
```

9.2.12 deleteCustomSPEngine

Online command that deletes a custom service provider (SP) integration engine from the configuration.

9.2.12.1 Description

This command deletes a custom SP integration engine in `config.xml`. The EngineID for an existing custom SP engine in `config.xml` must be provided.

9.2.12.2 Syntax

```
ddeleteCustomSPEngine('engineID')
```

Argument	Definition
engineID	Specifies the engine ID of an existing engine to be deleted.

9.2.12.3 Example

The following command deletes the engine with ID `id1234`.

```
wls:/mydomain/serverConfig> deleteCustomSPEngine('id1234')
```

9.2.13 deleteProviderFederation

Online command that deletes federations for given provider.

9.2.13.1 Description

This command deletes federations for given provider ID.

9.2.13.2 Syntax

```
deleteProviderFederation('providerID')
```

Argument	Definition
providerID	Specifies the ProviderID for the peer provider for which federation is to be deleted.

9.2.13.3 Example

The following command deletes `providerA`:

```
wls:/mydomain/serverConfig> deleteProviderFederation(providerA)
```

9.2.14 deleteUserFederations

Online command that deletes federations for given users.

9.2.14.1 Description

This command deletes federations for the given list of users.

9.2.14.2 Syntax

```
deleteUserFederations(['user1,..'])
```

Argument	Definition
user1	Specifies a comma-separated list of users whose federations are to be deleted. At least one user must be specified.

9.2.14.3 Example

The following command deletes federations for three users:

```
wls:/mydomain/serverConfig> deleteUserFederations(['userA','userB','userC'])
```

9.2.15 changeMessageStore

Online command that changes the message store between memory and RDBMS.

9.2.15.1 Description

This command changes the message store to memory or RDBMS.

9.2.15.2 Syntax

```
changeMessageStore('type','jndiname')
```

Argument	Definition
type	Specifies the type of store, RDBMS or Memory. Default is Memory.
jndiname	Specifies the `jndi` name to set for the store. Required if type is RDBMS.

9.2.15.3 Example

The following command changes the message store to RDBMS:

```
wls:/mydomain/serverConfig> changeMessageStore('RDBMS','jdbc/mydb')
```

9.2.16 changePeerProviderDescription

Online command that changes the peer provider description.

9.2.16.1 Description

This command updates a peer provider description in cot.xml.

9.2.16.2 Syntax

```
changePeerProviderDescription('providerID','description')
```

Argument	Definition
providerID	Specifies the provider ID.
description	Specifies the provider description.

9.2.16.3 Example

The following command updates the description of a provider:

```
wls:/mydomain/serverConfig> changePeerProviderDescription('providerA','new
description')
```

9.2.17 changeSessionStore

Online command that changes the session store between memory and RDBMS.

9.2.17.1 Description

This command changes the session store to memory or RDBMS.

9.2.17.2 Syntax

```
changeSessionStore('type','jndiname')
```

Argument	Definition
type	Specifies the type of store, RDBMS or Memory. Default is Memory.
jndiname	Specifies the jndi name to set for the store. Required if type is RDBMS.

9.2.17.3 Example

The following command changes the session store to RDBMS.

```
wls:/mydomain/serverConfig> changeSessionStore('RDBMS','jdbc/mydb')
```

9.2.18 createConfigPropertyList

Online command that creates a property list.

9.2.18.1 Description

This command creates a property list in config.xml.

9.2.18.2 Syntax

```
createConfigPropertyList('configName', 'listName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
listName	Specifies the property list name.

9.2.18.3 Example

The following command creates property list `mylistA`.

```
wls:/mydomain/serverConfig> createConfigPropertyList('serverconfig','mylistA')
```

9.2.19 createConfigPropertyListInMap

Online command that creates a property list nested in the property map.

9.2.19.1 Description

This command creates a property list, nested in the property map, in config.xml.

9.2.19.2 Syntax

```
createConfigPropertyListInMap('configName', 'mapName', 'listName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
mapName	Specifies an existing property map to contain the nested list.
listName	Specifies the property list name.

9.2.19.3 Example

The following command creates property list `mylistA` nested in a property map.

```
wls:/mydomain/serverConfig>
createConfigPropertyListInMap('serverconfig','mymapA','mylistA')
```

9.2.20 createConfigPropertyMap

Online command that creates a property map.

9.2.20.1 Description

This command that creates a property map in config.xml.

9.2.20.2 Syntax

```
createConfigPropertyMap('configName', 'mapName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
mapName	Specifies the property map to create.

9.2.20.3 Example

The following command creates property map `mymapA`:

```
wls:/mydomain/serverConfig> createConfigPropertyMap('serverconfig','mymapA')
```

9.2.21 createConfigPropertyMapInMap

Online command that creates a property map.

9.2.21.1 Description

This command that creates a property map in `config.xml`.

9.2.21.2 Syntax

```
ccreateConfigPropertyMapInMap('serverconfig','mymapA','nestedmapA')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
mapName	Specifies the name of an existing property map.
nestedMapName	Specifies the name of the property map to create nested inside mapName.

9.2.21.3 Example

The following command creates nested property map `nestedmymapA`:

```
wls:/mydomain/serverConfig>
createConfigPropertyMapInMap('serverconfig','mymapA','nestedmapA')
```

9.2.22 createFederationPropertyList

Online command that creates a property list.

9.2.22.1 Description

This command creates a property list in cot.xml.

9.2.22.2 Syntax

Argument	Definition
providerID	Specifies the provider ID.
listName	Specifies the name of the property list.

9.2.22.3 Example

The following command creates property list `mylistA`:

```
wls:/mydomain/serverConfig> createFederationPropertyList('providerA','mylistA')
```

9.2.23 createFederationPropertyListInMap

Online command that creates a property list nested in a property map.

9.2.23.1 Description

This command creates a property list, nested in a property map, in cot.xml.

9.2.23.2 Syntax

```
createFederationPropertyListInMap('providerID', 'mapName', 'listName')
```

Argument	Definition
providerID	Specifies the provider ID.
mapName	Specifies an existing property map to contain the nested list.
listName	Specifies the name of the property list.

9.2.23.3 Example

The following command creates nested property list mylistA:

```
wls:/mydomain/serverConfig>
createFederationPropertyListInMap('providerA','mymapA','mylistA')
```

9.2.24 createFederationPropertyMap

Online command that creates a property map.

9.2.24.1 Description

This command that creates a property map in cot.xml.

9.2.24.2 Syntax

```
createFederationPropertyMap('providerID', 'mapName')
```

Argument	Definition
providerID	Specifies the provider ID.
mapName	Specifies the name of the property map to be added to cot.xml.

9.2.24.3 Example

The following command creates property map `mymapA`:

```
wls:/mydomain/serverConfig> createFederationPropertyMap('providerA','mymapA')
```

9.2.25 createFederationPropertyMapInMap

Online command that creates a nested property map.

9.2.25.1 Description

This command creates a property map, nested in another property map, in cot.xml.

9.2.25.2 Syntax

```
createFederationPropertyMapInMap('providerID', 'mapName', 'nestedMapName')
```

Argument	Definition
providerID	Specifies the provider ID.
mapName	Specifies the name of an existing property map.
nestedMapName	Specifies the name of the property map to be nested inside mapName in cot.xml.

9.2.25.3 Example

The following command creates nested property map `nestedmapA`:

```
wls:/mydomain/serverConfig>
createFederationPropertyMapInMap('providerA','mymapA','nestedmapA')
```

9.2.26 createPeerProviderEntry

Online command that creates a peer provider property map entry.

9.2.26.1 Description

This command creates a peer provider as a Map property entry to cot.xml.

9.2.26.2 Syntax

```
createPeerProviderEntry('providerID', 'description', 'providerType','version')
```

Argument	Definition
providerID	Specifies the provider ID to be created.
description	This is the description of the provider ID.
providerType	Specifies the provider type of the peer provider to be created.
version	Specifies the version of the peer provider to be created.

9.2.26.3 Example

The following command creates a SAML 2.0 service provider:

```
wls:/mydomain/serverConfig> createPeerProviderEntry('providerA','idp
test','SP','SAML2.0')
```

9.2.27 getConfigListValueInMap

Online command that returns a list nested in a map.

9.2.27.1 Description

This command returns a list, nested in a map, from config.xml.

9.2.27.2 Syntax

```
getConfigListValueInMap('configName', 'mapName', 'listName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be accessed.
mapName	Specifies the name of the property map.
listName	Specifies the name of the list to be fetched from the map.

9.2.27.3 Example

The following command returns `mylistA`:

```
wls:/mydomain/serverConfig>
getConfigListValueInMap('serverConfig','mymapA','mylistA'
```

9.2.28 getConfigMapEntryInMap

Online command that returns a map property entry nested in a map.

9.2.28.1 Description

This command returns a map property entry, nested in a map, from config.xml.

9.2.28.2 Syntax

```
getConfigMapEntryInMap('configName', 'mapname', 'nestedMapName', 'propName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be accessed.
mapName	Specifies the name of the property map.
nestedmapName	Specifies the name of the nested property map.
propName	Specifies the name of the property to be fetched from the nested map.

9.2.28.3 Example

The following command returns property entry `myvarA`:

```
wls:/mydomain/serverConfig>
getConfigMapEntryInMap('serverconfig','mymap','nestedmapA','myvarA')
```

9.2.29 getConfigProperty

Online command that returns a property value.

9.2.29.1 Description

This command returns a property value from config.xml.

9.2.29.2 Syntax

```
getConfigProperty('configName', 'propName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be accessed.

Argument	Definition
propName	Specifies the name of the property to be fetched from the nested map.

9.2.29.3 Example

The following command returns property `myvarA`:

```
wls:/mydomain/serverConfig> getConfigProperty('serverconfig','myvarA')
```

9.2.30 getConfigPropertyList

Online command that returns a property list.

9.2.30.1 Description

This command returns a property list from config.xml.

9.2.30.2 Syntax

```
getConfigPropertyList('configName', 'listName')
```

Argument	Definition
configName	Specifies the configuration name.
listName	Specifies the name of the property list to be fetched from config.xml.

9.2.30.3 Example

The following command returns mylistA:

```
wls:/mydomain/serverConfig> getConfigPropertyList('serverconfig','mylistA')
```

9.2.31 getConfigPropertyMapEntry

Online command that returns a property value from a map.

9.2.31.1 Description

This command returns a property value from a map in config.xml.

9.2.31.2 Syntax

```
getConfigPropertyMapEntry('configName', 'mapName', 'propName')
```

Argument	Definition
configName	Specifies the configuration name (for example, idpsaml20, serverconfig, spsaml20,..).
mapName	Specifies the name of the property map.
propName	Specifies the name of the property to be fetched from the map in config.xml.

9.2.31.3 Example

The following command returns property `propA`:

```
wls:/mydomain/serverConfig> getConfigPropertyMapEntry('serverconfig','mapA',
'propA')
```

9.2.32 getFederationListValueInMap

Online command that returns a list value nested in a map.

9.2.32.1 Description

This command returns a list value nested in a map from cot.xml.

9.2.32.2 Syntax

```
getFederationListValueInMap('providerID', 'mapName', 'listName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map.
listName	Specifies the name of the list to be fetched from the map.

9.2.32.3 Example

The following command returns nested list mylistA:

```
wls:/mydomain/serverConfig>
getFederationListValueInMap('providerA','mymapA','mylistA')
```

9.2.33 getFederationMapEntryInMap

Online command that returns a map property entry nested in a map.

9.2.33.1 Description

This command returns a map property entry, nested in a map, from cot.xml.

9.2.33.2 Syntax

```
getFederationMapEntryInMap('providerID', 'mapname', 'nestedMapName', 'propName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map.
nestedmapName	Specifies the name of the nested property map.
propName	Specifies the name of the property to be fetched from the nested map.

9.2.33.3 Example

The following command returns property entry `myvarA`:

```
wls:/mydomain/serverConfig>
getFederationMapEntryInMap('providerA','mymap','nestedmapA','myvarA')
```

9.2.34 getFederationProperty

Online command that returns a property value.

9.2.34.1 Description

This command returns a property value from cot.xml.

9.2.34.2 Syntax

```
getFederationProperty('providerID', 'propName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
propName	Specifies the name of the property to be fetched from cot.xml.

9.2.34.3 Example

The following command returns property `myvarA`:

```
wls:/mydomain/serverConfig> getFederationProperty('providerA','myvarA')
```

9.2.35 getFederationPropertyList

Online command that returns a property list.

9.2.35.1 Description

This command returns a property list from cot.xml.

9.2.35.2 Syntax

```
getFederationPropertyList('providerID', 'listName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
listName	Specifies the name of the list to be fetched from the map.

9.2.35.3 Example

The following command returns list `mylistA`:

```
wls:/mydomain/serverConfig> getFederationPropertyList('providerA','mylistA')
```

9.2.36 extractproviderprops

Script command to export properties of a provider.

Note: This is a script command and cannot be executed directly from the online WLST prompt. Oracle Identity Federation scripts are located in `$ORACLE_HOME/fed/scripts`.

9.2.36.1 Description

Script command that extracts all the configuration properties of the specified provider and exports them to a text file. You can later use this file to set the same properties on another provider.

9.2.36.2 Syntax

Windows

```
IDM_ORACLE_HOME\common\bin\wlst.cmd extractproviderprops.py
providerID filename
```

Linux

```
IDM_ORACLE_HOME/common/bin/wlst.sh extractproviderprops.py
providerID filename
```

Argument	Definition
providerID	Specifies the name of the provider whose properties are to be extracted.
filename	Specifies the name of the text file to which the provider properties are extracted.

When you execute the script, you are prompted for the WebLogic administrator credentials and the connection URL; for the latter, specify the managed server port, not the admin server port.

File Format

The format of the extract file is:

```
TYPE:NAME:PROPNAME:PROPVALUE:PROPTYPE
```

For example:

```
X:X:sendattribute:false:boolean
MAP:attributelist/mailemail:datastore-attr:mail:string
LIST:sendattributefornameid:unspecified::string
```

9.2.37 setproviderprops

Script command to set properties of a provider using values from a text file.

Note: This is a script command and cannot be executed directly from the online WLST prompt. Oracle Identity Federation scripts are located in `$ORACLE_HOME/fed/scripts`.

9.2.37.1 Description

Script command to set properties of a provider using values from a text file.

The text file is generated by the "extractproviderprops" command.

9.2.37.2 Syntax

Windows

```
IDM_ORACLE_HOME\common\bin\wlst.cmd setproviderprops.py providerID filename
```

Linux

```
IDM_ORACLE_HOME/common/bin/wlst.sh setproviderprops.py providerID filename
```

Argument	Definition
providerID	Specifies the name of the provider whose properties are to be updated.
filename	Specifies the name of the input file from which to read the properties.

When you execute the script, you are prompted for the WebLogic administrator credentials and the connection URL; for the latter, specify the managed server port, not the admin server port.

9.2.38 getFederationPropertyMapEntry

Online command that returns a property value from a map.

9.2.38.1 Description

This command returns a property value from a map in cot.xml.

9.2.38.2 Syntax

```
getFederationPropertyMapEntry('providerID', 'mapName', 'propName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map.
propName	Specifies the name of the property to be fetched from the nested map.

9.2.38.3 Example

The following command returns property `propA` from a map:

```
wls:/mydomain/serverConfig> getFederationPropertyMapEntry('providerA','mapA',
'propA')
```

9.2.39 listCustomAuthnEngines

Online command that returns a list of custom authentication integration engines.

9.2.39.1 Description

This command returns a list of custom authentication integration engines from config.xml.

9.2.39.2 Syntax

```
listCustomAuthnEngines()
```

9.2.39.3 Example

The following command returns the list of all SP engines:

```
wls:/mydomain/serverConfig> listCustomAuthnEngines()
```

9.2.40 listCustomSPEngines

Online command that returns a list of custom SP integration engines.

9.2.40.1 Description

This command returns a list of custom service provider (SP) integration engines from config.xml.

9.2.40.2 Syntax

```
listCustomSPEngines()
```

9.2.40.3 Example

The following command returns the list of all SP integration engines:

```
wls:/mydomain/serverConfig> listCustomSPEngines()
```

9.2.41 loadMetadata

Online command that loads metadata from an input file.

9.2.41.1 Description

This command loads metadata from an input file into cot.xml.

9.2.41.2 Syntax

```
loadMetadata('metadatafile','description')
```

Argument	Definition
metadatafile	Specifies the metadata file of the peer provider to be added or updated.
description	This is a brief description of the peer provider to be loaded.

9.2.41.3 Example

The following command loads metadata from the file `metadatafile.xml`:

```
wls:/mydomain/serverConfig> loadMetadata('/home/metadatafile.xml','some
description')
```

9.2.42 oifStatus

Online command that reports the current status of the Oracle Identity Federation application in the managed server to which WLST is connected.

9.2.42.1 Description

This command displays the current status of Oracle Identity Federation on the managed server.

9.2.42.2 Syntax

```
oifStatus('serverurl', 'configfile', 'keyfile')
```

Argument	Definition
serverurl	Specifies the URL of the managed server.
configfile	This is a pre-defined user configuration file created with the WLST storeUserConfig command.
keyfile	This is a pre-defined key file created with the WLST storeUserConfig command

9.2.42.3 Example

The following command provides no arguments; WLST prompts you for the Oracle WebLogic Server username, password, and the managed server URL, then displays the federation server status:

```
wls:/mydomain/serverConfig> oifStatus()
```

The following command provides only the managed server URL; WLST prompts you for the Oracle WebLogic Server username and password:

```
wls:/mydomain/serverConfig> oifStatus('', '', 't3://localhost:7499')
```

The following command provides all arguments needed for WLST to display the federation server status:

```
wls:/mydomain/serverConfig> oifStatus('configfileA', 'keyfileB',
't3://localhost:7499')
```

9.2.43 removeConfigListInMap

Online command that removes a list property nested in a map.

9.2.43.1 Description

This command removes a list property nested in a map from config.xml.

9.2.43.2 Syntax

```
removeConfigListInMap('configName', 'mapName', 'listName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be accessed.
mapName	Specifies the name of the property map.
listName	Specifies the name of the list to be removed from the map.

9.2.43.3 Example

The following command removes the list property mylistA:

```
wls:/mydomain/serverConfig>
removeConfigListInMap('serverConfig','mymapA','mylistA')
```

9.2.44 removeConfigMapEntryInMap

Online command that removes a map property nested in a map.

9.2.44.1 Description

This command removes a map property entry nested in a map from config.xml.

9.2.44.2 Syntax

```
oifStatus('serverurl', 'configfile', 'keyfile')
```

Argument	Definition
serverurl	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be accessed.
configfile	Specifies the name of the property map.
keyfile	Specifies the name of the nested property map.

9.2.44.3 Example

The following command removes the nested property `myvarA`:

```
wls:/mydomain/serverConfig>
removeConfigMapEntryInMap('serverconfig','mymap','nestedmapA','myvarA')
```

9.2.45 removeConfigMapInMap

Online command that removes a map property nested in a map.

9.2.45.1 Description

This command removes a map property entry nested in a map from config.xml.

9.2.45.2 Syntax

```
removeConfigMapEntryInMap('configName', 'mapname', 'nestedMapName', 'propName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
mapName	Specifies the name of the property map.
nestedmapName	Specifies the name of the nested property map.
propName	Specifies the name of the property to be removed from the nested map.

9.2.45.3 Example

The following command removes the nested property myvarA:

```
wls:/mydomain/serverConfig>
removeConfigMapEntryInMap('serverconfig','mymap','nestedmapA','myvarA')
```

9.2.46 removeConfigProperty

Online command that removes a configuration property.

9.2.46.1 Description

This command removes a property from config.xml.

9.2.46.2 Syntax

```
removeConfigProperty('configName', 'propName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
propName	Specifies the name of the property to be removed.

9.2.46.3 Example

The following command removes the property `myvarA`:

```
wls:/mydomain/serverConfig> removeConfigProperty('serverconfig','myvarA')
```

9.2.47 removeConfigPropertyList

Online command that removes a configuration property list.

9.2.47.1 Description

This command removes a property list from config.xml.

9.2.47.2 Syntax

```
removeConfigPropertyList('configName', 'listName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
listName	Specifies the name of the property list to be removed.

9.2.47.3 Example

The following command removes the property list `mylistA`:

```
wls:/mydomain/serverConfig> removeConfigPropertyList('serverconfig','mylistA')
```

9.2.48 removeConfigPropertyMap

Online command that removes a property map.

9.2.48.1 Description

This command removes a property map in config.xml.

9.2.48.2 Syntax

```
removeConfigPropertyMap('configName', 'mapName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
mapName	Specifies the name of the property map to be removed.

9.2.48.3 Example

The following command removes mapA:

```
wls:/mydomain/serverConfig> removeConfigPropertyMap('serverconfig','mapA')
```

9.2.49 removeConfigPropertyMapEntry

Online command that removes a property value from a map.

9.2.49.1 Description

This command removes a property value from a map in config.xml.

9.2.49.2 Syntax

```
removeConfigPropertyMapEntry('configName', 'mapName', 'propName')
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.

Argument	Definition
mapName	Specifies the name of the property map to be updated.
propName	Specifies the name of the property to be removed from the map.

9.2.49.3 Example

The following command removes property propA:

```
wls:/mydomain/serverConfig> removeConfigPropertyMapEntry('serverconfig','mapA',
'propA')
```

9.2.50 removeFederationListInMap

Online command that removes a property list in a map.

9.2.50.1 Description

This command removes a property list in a map, in cot.xml.

9.2.50.2 Syntax

```
removeFederationListInMap('providerID', 'mapName', 'listName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map.
listName	Specifies the name of the property list to be removed.

9.2.50.3 Example

The following command removes mylistA in mymapA:

```
wls:/mydomain/serverConfig>
removeFederationListInMap('providerA','mymapA','mylistA')
```

9.2.51 removeFederationMapInMap

Online command that removes a nested map in a map.

9.2.51.1 Description

This command removes a property map nested inside a map in cot.xml.

9.2.51.2 Syntax

```
removeFederationMapInMap('providerID', 'mapname', 'nestedMapName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map containing the nested map.
nestedmapName	Specifies the name of the nested property map to be removed.

9.2.51.3 Example

The following command removes `nestedmapA` in `mymap`:

```
wls:/mydomain/serverConfig>
removeFederationMapInMap('providerA','mymap','nestedmapA')
```

9.2.52 removeFederationMapEntryInMap

Online command that removes a nested map property entry in a map.

9.2.52.1 Description

This command removes a property name/value pair to a map nested inside a map in cot.xml.

9.2.52.2 Syntax

```
removeFederationMapEntryInMap('providerID', 'mapname', 'nestedMapName',
'propName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map containing the nested map.
nestedmapName	Specifies the name of the nested property map.
propName	Specifies the name of the property to be removed from the nested map.

9.2.52.3 Example

The following command removes map property entry myvarA:

```
wls:/mydomain/serverConfig>
removeFederationMapEntryInMap('providerA','mymap','nestedmapA','myvarA')
```

9.2.53 removeFederationProperty

Online command that removes a property value.

9.2.53.1 Description

This command removes a property entry in cot.xml.

9.2.53.2 Syntax

```
removeFederationProperty('providerID', 'propName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be updated.
propName	Specifies the name of the property to be removed.

9.2.53.3 Example

The following command removes the provider property myvarA:

```
wls:/mydomain/serverConfig> removeFederationProperty('providerA','myvarA')
```

9.2.54 removeFederationPropertyList

Online command that removes a property list entry.

9.2.54.1 Description

This command removes a property list entry in cot.xml.

9.2.54.2 Syntax

```
removeFederationPropertyList('providerID', 'listName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
listName	Specifies the name of the property list to be removed.

9.2.54.3 Example

The following command removes `mylistA`:

```
wls:/mydomain/serverConfig> removeFederationPropertyList('providerA','mylistA')
```

9.2.55 removeFederationPropertyMap

Online command that removes a property map.

9.2.55.1 Description

This command removes a property map in cot.xml.

9.2.55.2 Syntax

```
removeFederationPropertyMap('providerID', 'mapName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map to be removed.

9.2.55.3 Example

The following command removes a map:

```
wls:/mydomain/serverConfig> removeFederationPropertyMap('providerA','mapA')
```

9.2.56 removeFederationPropertyMapEntry

Online command that removes a property value from a map.

9.2.56.1 Description

This command removes a property value from a map in cot.xml.

9.2.56.2 Syntax

```
removeFederationPropertyMapEntry('providerID', 'mapName', 'propName')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map to be updated.
propName	Specifies the name of the property to be removed from the map.

9.2.56.3 Example

The following command removes property `propA` from a map:

```
wls:/mydomain/serverConfig> removeFederationPropertyMapEntry('providerA','mapA',
'propA')
```

9.2.57 removePeerProviderEntry

Online command that removes a peer provider entry.

9.2.57.1 Description

This command removes a peer provider entry from cot.xml.

9.2.57.2 Syntax

```
removePeerProviderEntry('providerID')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be removed.

9.2.57.3 Example

The following command removes providerA:

```
wls:/mydomain/serverConfig> removePeerProviderEntry('providerA')
```

9.2.58 setConfigProperty

Online command that sets a property value in config.xml.

9.2.58.1 Description

This command adds or updates a property value in config.xml.

9.2.58.2 Syntax

```
setConfigProperty('configname', 'propName', 'value', 'type')
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
propname	Specifies the name of the property to be added/updated in config.xml.
value	Specifies the property value.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.58.3 Example

The following command sets the property `myvarA` and its value in the server configuration:

```
wls:/mydomain/serverConfig>
setConfigProperty('serverconfig','myvarA','myvalA','string')
```

9.2.59 setCustomAuthnEngine

Online command that updates a custom authentication integration engine.

9.2.59.1 Description

This command updates a custom authentication integration engine in config.xml.

9.2.59.2 Syntax

```
setCustomAuthnEngine('engineID', 'name', 'enabled', 'webContext',
'authnRelativePath', 'logoutRelativePath', 'logoutEnabled')
```

Argument	Definition
engineID	Specifies the engine ID of an existing engine.
name	Specifies the name of the custom engine.
enabled	This flag specifies whether the engine is enabled (true) or not (false).
webContext	Specifies the web context for the engine.
authnRelativePath	Specifies the authentication relative path URL for the engine.
logoutRelativePath	Specifies the logout relative path URL for the engine.
logoutEnabled	This flag is set true to enable logout for the engine, else false.

9.2.59.3 Example

The following command updates the configuration of custom authentication engine `abcdef`:

```
wls:/mydomain/serverConfig> setCustomAuthnEngine('abcdef',
'custom one','false','oracle:fed:authentication:unspecified','webcontext')
```

9.2.60 setCustomSPEngine

Online command that updates a custom SP integration engine.

9.2.60.1 Description

This command updates an existing custom SP integration engine in config.xml.

9.2.60.2 Syntax

```
setCustomSPEngine('engineID' ,'name', 'enabled', 'authnMech', 'webContext',
'authnRelativePath', 'logoutRelativePath', 'logoutEnabled')
```

Argument	Definition
engineID	Specifies the engine ID of an existing custom engine.
name	Specifies the name of the custom engine.
enabled	This flag specifies whether the engine is enabled (true) or not (false).
authnMech	Specifies the authentication mechanism for the engine.
webContext	Specifies the web context for the engine.
authnRelativePath	Specifies the authentication relative path URL for the engine.
logoutRelativePath	Specifies the logout relative path URL for the engine.
logoutEnabled	This flag is set true to enable logout for the engine, else false.

9.2.60.3 Example

The following command sets the name and the enabled flag for the engine with ID `engineID2`:

```
wls:/mydomain/serverConfig> setCustomSPEngine('engineid2','test','true')
```

9.2.61 setFederationProperty

Online command that adds or updates a property value.

9.2.61.1 Description

This command adds a property entry or updates an existing entry in cot.xml.

9.2.61.2 Syntax

```
setFederationProperty('providerID', 'propName', 'value', 'type')
```

Argument	Definition
providerID	Specifies the name of the peer provider to be updated.
propname	Specifies the name of the property to be added/updated in cot.xml.
value	Specifies the property value.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

9.2.61.3 Example

The following command creates the property `myvarA` and sets its value:

```
wls:/mydomain/serverConfig>
setFederationProperty('providerA','myvarA','myvalA','string')
```

Part III

Oracle Universal Federation Framework

This part explains how to use Oracle Universal Federation Framework to customize Oracle Identity Federation.

Part III contains the following chapters:

10 Integrating with Third-Party Identity and Access Management Modules

This section includes the following topics related to custom engine implementation:

10.1 Background for Custom Implementations

Out of the box, Oracle Identity Federation integrates with several Identity and Access Management (IAM) products, including Oracle Access Manager, Oracle AS Single Sign-On, and others. See Section 5.15, "Configuring Authentication Engines" and Section 5.16, "Configuring SP Integration Modules" for details.

See Also: Section 1.2.2, "Architecture" for a discussion of Oracle Identity Federation and its relationship to other federation components.

This section explains the components of the framework for custom IAM products, and shows you how to configure and integrate a custom IAM solution into the framework.

Note: Oracle strongly discourages users from deploying any applications on the Oracle WebLogic Managed Server other than the custom integration and authentication described in this section, because doing so introduces potential security risks. Extraneous applications deployed in the Oracle WebLogic Managed Server can potentially affect the security of the federation server by allowing rogue software to change the behavior of the server flows.

10.2 Architecture and Flows

At runtime, Oracle Identity Federation interacts with two types of external modules: a user data store, and an Identity and Access Management (IAM) system.

Oracle Identity Federation works with the user data store to:

- locate a user after local authentication

- locate a user after processing an incoming SAML assertion
- retrieve attributes for a specific user

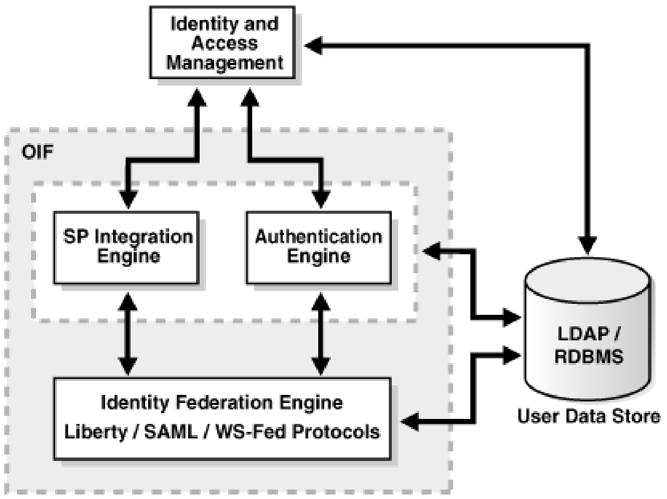
The Identity and Access Management (IAM) system provides access control for protected resources. Oracle Identity Federation, as the federation server, interacts with IAM to:

- authenticate a user when the server needs to obtain a user's local identity. This operation might occur when the server acts as an IdP, or when the server, as an SP, needs to authenticate the user during initial account linking/federation creation.
- create an authenticated session for a specific user when the server processes an incoming SAML assertion and asserts the user's identity to the IAM system
- process logout flows; in this case, the federation server invokes IAM capabilities to log the user out of the system

10.2.1 Architecture

Figure 3–1 depicts the different external and internal modules of an Oracle Identity Federation deployment, and how they interact at runtime:

Figure 10–1 Oracle Identity Federation Module Interactions


Identity and Access Management
OIF
SP Integration Engine
Authentication Engine
Identity Federation Engine
Liberty / SAML / WS-Fed Protocols
LDAP / RDBMS
User Data Store

For the sake of this discussion, Oracle Identity Federation is depicted as three internal modules:

- the Identity Federation engine, which is responsible for creating and processing SAML messages such as AuthnRequest, assertion, and logout messages.

  This module:

  – works with the user data store when processing SAML messages;

  – interacts with the authentication engine when it is necessary for a user to be locally identified; and

  – interacts with the SP integration engines when the user is redirected to the IAM component after processing an assertion.

- the authentication engines, which are responsible for processing requests from the federation engine to authenticate users. This module interacts with the IAM component and the user data store to authenticate the user and retrieve the unique identifier Oracle Identity Federation uses to reference the user.

This module can be invoked in either IdP or SP mode when local authentication is required.

After authenticating the user, the authentication engine sends the authentication information, such as the user's identifier, the time of authentication, and other data to the federation engine.

- After successfully processing an incoming SAML assertion and locating the user referenced by the assertion, the federation engine instructs the SP integration engine to create an authenticated session for that user in the IAM domain. It passes the necessary information (user's identifier, authentication time, and so on) to the SP integration engine, which interacts with the IAM server to create the session.

10.2.2 Authentication Engine Framework

The authentication engine framework of Oracle Identity Federation is designed to authenticate a user, and includes several internal plug-ins that allow it to interact with the various IAM servers supported out of the box, such as:

- Oracle Single Sign-On
- Oracle Access Manager
- Username/Password using an LDAP server
- Username/Password using an RDBMS server
- Username/Password using an RDBMS Table
- InfoCard
- Federation SSO Proxy
- JAAS

Additionally, Oracle Identity Federation provides a framework so that the server can be integrated with third party authentication modules. Integrators have two approaches to customize Oracle Identity Federation to interact with other authentication platforms:

- Design and implement an authentication module that will interact with Oracle Identity Federation using internal JavaEE Servlet forwards. The new module would also be linked to the third party authentication solution responsible of authenticating the user.
- Leverage JAAS to use a specific Login module. Oracle Identity Federation includes an authentication engine, based on internal forwards, which uses the JAAS libraries to authenticate the user. This allows integrators to re-use any JAAS compliant login modules that already exist for use in Oracle Identity Federation server modules by configuring Oracle Identity Federation.

Here is a step-by-step description of how the authentication engine interacts with other components in a typical user flow (Oracle Identity Federation is also referred to as the federation server):

1. The user accesses Oracle Identity Federation for an SSO operation (in either SP or IdP mode).

2. An internal process in the server determines that the user needs to be identified using a specific authentication mechanism (either the default one from the configuration or one requested by a remote service provider).
3. The federation server determines which authentication engine to use to challenge/identify the user for the specified authentication mechanism.
4. The federation server then internally forwards the user's request to the Web Context and Login Relative Path of the authentication engine to challenge/identify the user, and it passes some information, specified via Java Objects stored as Attributes of the `HttpServletRequest` instance:
   - The authentication mechanism to use when challenging the user for identification
   - An identifier referencing the current action that is being performed
   - The `ProviderID` and the description of the remote service provider for which this local authentication is requested, if a Federation SSO operation is performed
   - The identifier referencing the engine used to authenticate the user
   - The identifier of the user
   - The `Force Authentication` flag, indicating whether the engine should challenge the user even if the user is already authenticated.
   - The `Is Passive` flag, indicating whether the engine is allowed to visually interact with the user.
   - Optionally, a map of attributes that need to be set by the engine: these attributes are required in order for Oracle Identity Federation/IdP to create correctly the assertion with the `AttributeStatement`, as specified by the configuration for that specific remote provider.
   - Optionally, a `String` containing the Oracle Identity Federation session identifier, if the user has already an active session. Oracle Identity Federation is passing the sessionID of the already existing user session (if one exists), to the authentication engine, so that the engine can persist state linked to the user, and it can reference that data by using the `sessionID` value. Later on, when the logout flow is being executed, Oracle Identity Federation passes the `sessionID` that is being logged out to the engine, so that the engine can delete the data that was used for this user session.
5. The authentication engine processes the incoming request, and it has access to the information stored as `HttpServletRequest` attributes.
6. The authentication engine interacts with the IAM component and may challenge the user for credentials. After successful authentication, it may set a cookie (for example, to maintain the authenticated session with the IAM server and/or the target application).
7. The authentication module sends the user back to Oracle Identity Federation using an internal forward (Web Context `/fed` and Login Relative Path `/user/loginsso`), and it passes the following information as `HttpServletRequest` attributes:
   - The identifier of the user
   - Authentication time
   - Expiration time of the authenticated session
   - The authentication mechanism used to identify the user

- The identifier referencing the action that was being performed, from the request
- The identifier referencing the engine used to authenticate the user
- Optionally, a map of attributes that is stored in the user session.
- Optionally, a String containing the Oracle Identity Federation session identifier that Oracle Identity Federation needs to use to reference the Oracle Identity Federation user session. This allows the engine and Oracle Identity Federation to share the same identifier to reference the user session. Later on, when the logout flow is being executed, Oracle Identity Federation passes the `sessionID` that is being logged out to the engine, so that the engine can delete the data that was used for this user session.

Note:

- If the user ID attribute is empty but the authentication time and authentication mechanism attributes are not empty, it tells Oracle Identity Federation that the authentication succeeded, but that the user is unknown on the server. This is useful when Oracle Identity Federation, acting as an IdP, is configured to use the attributes passed by the engine to create an assertion
- If the authentication time or authentication mechanism attributes are empty, it tells Oracle Identity Federation that the authentication failed.

8. Oracle Identity Federation performs these actions:
   - processes the incoming request
   - retrieves the data embedded as attributes in the `HttpServletRequest`
   - locates the user in the user data store
   - creates a session for the user
   - sets a cookie, and
   - resumes the SSO operation.

10.2.3 SP Integration Engine Framework

The SP integration engine included with Oracle Identity Federation consists of a servlet that processes requests from the server to create a user authenticated session at the IAM server. The engine includes several internal plug-ins that allow it to interact with different IAM servers, such as:

- Oracle Single Sign-On
- Oracle Access Manager
- Oracle Identity Federation Test Application

Additionally, Oracle Identity Federation provides a framework so that the server is able to be integrated with third party IAM frameworks: the customized SP integration Module interacts with Oracle Identity Federation using internal J2EE Servlet forwards, and it communicates with the third party IAM system to create the user authenticated session.

Here is a step-by-step description of how an SP integration engine interacts with the Oracle Identity Federation Framework in a typical user flow:

1. The user attempts to access a resource protected by the IAM solution, and configured to use Federation SSO to authenticate the user.
2. The IAM deployment redirects the user to the corresponding SP integration Module on Oracle Identity Federation.
3. The SP integration Module decodes the information sent by the IAM deployment and internally forward the user to the Oracle Identity Federation server with the following information set as `HttpServletRequest` attributes:
   - An optional authentication mechanism specifying to the SP which authentication mechanism to request the IdP to use during authentication.

     Note: if set, this parameter is used to determine the IdP to use, disregarding the default parameter described next.

   - An optional Provider ID referencing the IdP to use for the Federation SSO. If missing, Oracle Identity Federation uses the IdP mapped for the specified authentication mechanism. If no IdP could be found, Oracle Identity Federation uses the IdP configured as the Default SSO IdP
   - An optional federation ID referencing the affiliation to use to trigger the Federation SSO
   - The relay state. It can contain a small string, for example a reference to some data saved in a repository or a small URL pointing to the protected resource to redirect the user to after completion of the SSO operation
   - The identifier of the SP engine that started the SSO flow
   - An optional boolean indicating if the Oracle Identity Federation server should authenticate the user locally using the authentication engines or if a Federation SSO should be started by redirecting the user to an IdP for authentication
   - A `Boolean` object indicating whether to use the configuration stored in Oracle Identity Federation or to only start the SSO operation based on the information being passed by the SP engine, except for the IdP
   - A `Boolean` object indicating whether the SP should ask the IdP to challenge the user even if already authenticated
   - A `Boolean` object indicating whether the SP should allow the IdP to create a federation record if one does not yet exist, during the SSO operation
   - A `Boolean` object indicating whether the SP should ask the IdP not to interact with the user during the SSO operation
   - A `String` representing the binding to use when sending the `AuthnRequest`
   - A `String` representing the binding to use when sending the response with the assertion
   - An optional authentication mechanism comparison specifying to the SP which authentication context comparison to request the IdP to use during authentication
   - A `String` representing the NameID format the SP uses to ask the IdP for the SSO operation

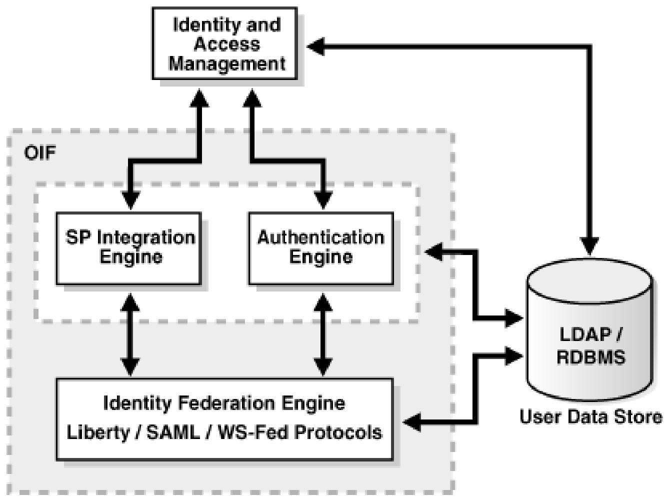
4. Oracle Identity Federation initiates a Federation SSO operation with a remote IdP.
5. The IdP authenticates the user and, if necessary, redirects the user, with an assertion, to the federation server acting as an SP.
6. The server processes the assertion and locates the user in the user data store. The user is now authenticated at the federation server.
7. Oracle Identity Federation internally forwards the user back to the SP integration Module by using the Web Context and Login Relative Path of that module configured in Oracle Identity Federation. The server passes the following data as `HttpServletRequest` attributes:
   - A `Boolean` object indicating if the SSO operation was successful
   - The identifier of the user
   - Authentication time
   - Expiration time of the authenticated session
   - The authentication mechanism used to identify the user
   - The relay state
   - The contents of the assertion: the `NameID`, the `Issuer` of the assertion and the optional attributes. Note: the content of the assertion is not passed as XML Data, that is the original assertion will not be passed back to the module. The extra data is referenced as:
     – `orafed-nameid-value` containing the Name ID value
     – `orafed-nameid-qualifier` containing the Name ID qualifier
     – `orafed-nameid-format` containing the Name ID format
     – `orafed-providerid` containing the Peer ProviderID
   - The top status of the SAML Response
   - The low status of the SAML Response if any
   - The status message if any
   - The `ProviderID` that created the SSO assertion
   - The identifier of the SP engine to process the above information
   - A `String` containing the Oracle Identity Federation identifier of the user session. Oracle Identity Federation is passing the sessionID of the user session to the SP engine, so that it can persist state linked to the user, and it can reference that data by using the `sessionID` value. Later on, when the logout flow is being executed, Oracle Identity Federation passes the sessionID that is being logged out to the engine, so that the engine can delete the data that was used for this user session.
8. The SP integration engine interacts with the IAM server to create an authenticated session for the user. The session is based on the data received from Oracle Identity Federation.
9. The SP integration engine redirects the user to the final target URL.

10.2.4 Logout

When logging out, Oracle Identity Federation and the authentication/SP engines need to be logged out. This involves:

1. Logging out the user from the authentication engines
2. Logging out the user from the SP engines
3. Performing the SAML/WS-Fed Global Logout profiles
4. Logging the user out from Oracle Identity Federation

Figure 10–2 Oracle Identity Federation Module Interactions


Identity and Access Management
OIF
SP Integration Engine
Authentication Engine
Identity Federation Engine
Liberty / SAML / WS-Fed Protocols
LDAP / RDBMS
User Data Store

There are several ways to invoke the logout:

- The user invokes the Oracle Identity Federation logout server, at `/fed/user/logout` by specifying an optional return URL. In this case, Oracle Identity Federation logs the user out from authentication/SP engines, the remote SAML providers and from Oracle Identity Federation itself, and Oracle Identity Federation redirects the user to the return URL, or display the logout result page.
- The user is redirected from a remote SAML/WS-Fed provider to Oracle Identity Federation using the Global Logout protocol. In this case, Oracle Identity Federation logs the user out from authentication/SP engines, the remote SAML/WS-Fed providers (except the one that sent the logout message), from Oracle Identity Federation itself and redirect the user back to the remote SAML provider that sent the original message.
- The user initiates logout from an environment integrated with an authentication/SP engine. In that case, that environment would invoke the authentication/SP engine for logout, and the engine would then send the user to Oracle Identity Federation for logout. From that point, Oracle Identity Federation would log out the user from the authentication/SP engines (except the engine that redirected the user to Oracle Identity Federation), from Oracle Identity Federation itself and redirect the user back to the authentication/SP engine that started the flow

Note: Internal forwards is used to send the user from Oracle Identity Federation to the authentication/SP engines and from the authentication/SP engines to Oracle Identity Federation.

Oracle Identity Federation invokes Authn/SP Engine

When Oracle Identity Federation sends the user to the authentication/SP engine, it:

1. Performs an internal forward to the web context and relative logout path of the engine
2. Specifies the engine ID of the invoked engine for logout
3. Optionally specifies the identifier of the user session being logged out

When the authentication/SP engine logs the user out, the engine internally forwards the user back to Oracle Identity Federation, it:

1. Performs an internal forward to the /fed web context and /user/logoutretsso
2. Specifies the engine ID of the invoked engine

Authn/SP invokes Oracle Identity Federation

When an authentication/SP engine invokes Oracle Identity Federation for logout, it:

1. Performs an internal forward to the /fed web context and /user/logoutsso
2. Specifies the engine ID of the invoked engine for logout
3. Specifies a return URL where Oracle Identity Federation redirects the user after logout.

At the end of the logout flow, the user is logged out from Oracle Identity Federation and redirected to the return URL.

10.2.5 Requirements

Oracle Identity Federation's design is consistent with certain requirements for authentication operations and SP integration (where a user session is created at the IAM server). Consequently, you must meet the following requirements when implementing a custom authentication engine or an SP integration engine:

- The authentication engine, the SP integration engine, the Oracle Identity Federation engine and the IAM server must use the same user data store as the user repository. This store contains the user data used to look up and authenticate users.
- The authentication engine and the SP integration engine must include a Java Servlet /JSP.
- The data exchanges between Oracle Identity Federation and the authentication/SP integration engines are done via internal HTTP request forwarding. This is actually an internal API call between the modules that relies on the J2EE servlet framework via the HTTP protocol.
- A logout service needs to be implemented and made available to the authentication engine and/or the SP integration engine. This logout service must be published as Servlet/JSP.

10.3 Creating a Custom Authentication Engine

This section explains how to plan, develop, and implement a custom authentication engine.

10.3.1 Planning a Custom Authentication Engine

Creating a customized authentication engine involves:

- creating a service that will process incoming requests from Oracle Identity Federation

- implementing a module to authenticate a user
- creating a service that forwards the user to the federation server with the required information
- deciding whether the authentication engine will set a cookie after authenticating a user. If yes, the authentication module must be integrated into the logout process (see Section 10.5, "Logout")
- packaging the services and module into a web application, and deploying the application to the Oracle WebLogic Managed Server where Oracle Identity Federation is running
- configuring Oracle Identity Federation to reference the new authentication engine.
- ensuring that the user identifier returned by the authentication engine references the same user in the Oracle Identity Federation User Data Store

10.3.2 Developing and Implementing the Authentication Module

Several aspects of module development are explained here.

URLs

Communication between the federation engine and the authentication engine occurs through internal servlet forwards that are equivalent to API calls. These forwards use the following JavaEE API:

```
ServletContext.getContext(String contextPath)
   .getRequestDispatcher(String relativePath)
   .forward(HttpServletRequest request,
      HttpServletResponse response)
```

where:

- `contextPath` is the root context path of the web application. For example, the `contextPath` of Oracle Identity Federation is `/fed`.
- `relativePath` is the service URL to which to forward the user; it is relative to the `contextPath`. For example, after authenticating a user, the authentication engine uses `/user/loginsso` as the `relativePath` when forwarding the user.

Oracle Identity Federation needs to be aware of the ID of the new authentication engine, and the `contextPath` and the `relativePath`. This is the URL that will process authentication requests issued by the federation server.

Adding or Modifying an Authentication Engine

To add an authentication engine or modify one:

- Go to Fusion Middleware Control and locate the Oracle Identity Federation instance.
- Navigate to **Administration** then **Authentication Engines**.
- To add an authentication engine, click **Add** and enter a name for that authentication engine. Oracle Identity Federation generates an ID for that new engine.

Note: Fusion Middleware Control only uses the name for display purposes while the ID is used during communications between Oracle Identity Federation and the authentication engine.

- To modify an authentication engine, select it and:
  - Enable or disable the engine
  - Specify the `contextPath` of the authentication engine in the Web Context field
  - Specify the relative path of the login service of the authentication engine in the Login Relative Path field
  - Enable or disable logout
  - Specify the relative path of the logout service of the authentication engine in the **Logout Relative Path** field
- Click **Apply**.

See Also: Section 5.15, "Configuring Authentication Engines"

Implementing the Service

This section describes the roles that are played by the authentication engine, and the processing tasks that the service must be able to handle for a successful implementation.

The authentication engine needs to:

- process requests from the federation engine
- forward the user to the federation server after a successful authentication

When processing authentication requests from the server, the engine must process the following incoming data:

- The authentication mechanism to use when challenging the user for identification as a String (identified by oracle.security.fed.authn.authnmech)
- An identifier referencing the current action that is being performed as a String (identified by `oracle.security.fed.authn.refid`)
- The `ProviderID` and the description of the remote service provider for which this local authentication is requested, if a Federation SSO operation is performed as a String (identified by `oracle.security.fed.authn.providerid` and `oracle.security.fed.authn.providerdescription`).
- The identifier referencing the engine used to authenticate the user (identified by oracle.security.fed.authn.engineid)
- The identifier of the user as a `String`, if set (identified by oracle.security.fed.authn.userid)
- The Force Authentication flag, a `Boolean` object, indicating whether the engine should challenge the user even if the user is already authenticated. If missing, False is assumed. (identified by `oracle.security.fed.authn.forceauthn`)
- The `Is Passive` flag, a `Boolean` object, indicating whether the engine is allowed to visually interact with the user. If missing, `False` is assumed. (identified by `oracle.security.fed.authn.passive`)
- Optionally, a map of attributes that need to be set by the engine: these attributes are required for Oracle Identity Federation/IdP to correctly create the assertion with the AttributeStatement, as specified by the configuration for that specific remote provider. (identified by `oracle.security.fed.authn.attributes`)

  When Oracle Identity Federation receives an SSO assertion, processes it and requests that the user be locally authenticated because the server was not able to map the assertion to a local user, the Map contains this data from the assertion:

  - `orafed-nameid-value` – the user's Name ID value
  - `orafed-nameid-qualifier` – the user's Name ID qualifier
  - `orafed-nameid-format` – the user's Name ID format
  - `orafed-providerid` – the IdP's ProviderID
  - `orafed-assertionid` - the ID of the assertion
  - `orafed-xmlmessage` - the optional XML message containing the assertion.

  See Section 6.13.2, "Providing XML Message to SP Engine after SSO Completes" for details.

- Optionally, a `String` containing the Oracle Identity Federation session identifier, if the user has already an active session. Oracle Identity Federation is passing the sessionID of the already existing user session (if one exists), to the authentication engine, so that the engine can persist state linked to the user, and it can reference that data by using the sessionID value. Later on, when the logout flow is being executed, Oracle Identity Federation will pass the sessionID that is being logged out to the engine, so that the engine can delete the data that was used for this user session. (identified by `oracle.security.fed.sessionid`).

  See Also: Section 5.9.2.1, "Configuring Attribute Name Mapping".

After successful authentication, the engine must forward the user to the federation server with the `rootContext` of the federation engine being `/fed`, and the relativePath `/user/loginsso`.

Oracle Identity Federation expects this data when processing the internal forward:

- The identifier of the user as a `String` (identified by `oracle.security.fed.authn.userid`)
- Authentication time as a `Date` object (identified by `oracle.security.fed.authn.authntime`)
- Expiration time of the authenticated session as a `Date` object (identified by `oracle.security.fed.authn.expirationtime`)
- The authentication mechanism used to identify the user as a String (identified by `oracle.security.fed.authn.authnmech`)
- The identifier referencing the action that was being performed, from the request (identified by `oracle.security.fed.authn.refid`)
- The identifier referencing the engine used to authenticate the user (identified by `oracle.security.fed.authn.engineid`)
- Optionally, a Map of attributes that is stored in the user session. This map will have String objects as the keys and a set of objects as the values (identified by `oracle.security.fed.authn.attributes`).
- Optionally, a `String` containing the Oracle Identity Federation session identifier that Oracle Identity Federation will need to use to reference the Oracle Identity Federation user session. This allows the engine and Oracle Identity Federation to share the same identifier to reference the user session. Later on, when the logout flow is being executed, Oracle Identity Federation will pass the sessionID that is being logged out to the engine, so that the engine can delete the data that was used for this user session. (identified by `oracle.security.fed.sessionid)`

Notes:

- If the oracle.security.fed.authn.userid attribute is empty but the `oracle.security.fed.authn.authntime` and `oracle.security.fed.authn.authnmech` attributes are not empty, it tells Oracle Identity Federation that the authentication succeeded, but that the user is unknown on the server. This is useful when Oracle Identity Federation, acting as an IdP, is configured to use the attributes passed by the engine to create an assertion.

  If the oracle.security.fed.authn.userid attribute is null, the IdP must be configured to not use any federation data stores, since the assertion data will be solely based on information passed from the custom authentication engine.
- Use the XML-based federation store only for testing, and not in a production environment.
- If the `oracle.security.fed.authn.authntime` or `oracle.security.fed.authn.authnmech` attributes are empty, it tells Oracle Identity Federation that the authentication failed

Here are some additional implementation requirements:

- If the service needs to set any cookies, perform this operation before forwarding the user to the federation server.
- Set the cookie path value to "/". This is required because of the internal forwards between the Oracle Identity Federation web application and the authentication engine web application; the user's browser needs to send the cookies related to the authentication engine, even when it is accessing only the federation server. This way, at an internal forward from the federation server to the authentication engine, the cookies set by the engine are available in the HTTP Request.

10.3.3 Sample Authentication Module for Oracle Single Sign-On Integration

This section describes how to integrate a custom authentication engine with Oracle Single Sign-On.

Setup

In this example, the application server where Oracle Identity Federation is running has been integrated with the Oracle Single Sign-On server, and the SSO module statically protects the `/engine/forward.jsp` URL.

Additionally, the user data store configured for Oracle Identity Federation references the Oracle Internet Directory server used by Oracle Single Sign-On.

See Also: Section 3.2.2, "Deploying Oracle Identity Federation with Oracle Single Sign-On" for more information on SSO integration.

Packaging

The authentication engine consists of a Web application with a root context set to `/engine`, and contains two JSP pages:

- `authentication.jsp` which processes the incoming request from Oracle Identity Federation

- `forward.jsp` which is protected by Oracle Single Sign-On, and which forwards the user back to the federation server with the required data.

Adding the Engine

To add the engine:

- Go to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
- Navigate to **Administration**, then **Authentication Engines**. Select the **Authentication Engines - Custom** tab.
- To add an authentication engine, click **Add** and enter a name for that authentication engine. Oracle Identity Federation will generate an ID for that new engine: this ID is reference by TEST_ENGINE_ID for this test
- Select the authentication engine to modify it:
  - Enable the engine.
  - Set "/engine" as the Web Context of the authentication engine
  - Set "/authentication.jsp" as the Login Relative Path of the authentication engine
- Click **Apply**.

See Also: Section 5.15, "Configuring Authentication Engines"

Implementation of authentication.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.net.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

String authnMech =
(String)request.getAttribute("oracle.security.fed.authn.authnmech");
String refid = (String)request.getAttribute("oracle.security.fed.authn.refid");

String redirectURL = "/engine/forward.jsp?refid=" +
   (refid != null ? URLEncoder.encode(refid) : "");

response.sendRedirect(redirectURL);
%>
```

Implementation of forward.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.util.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

String refid = request.getParameter("refid");
String userID = request.getRemoteUser();
String authnMethod = "oracle:fed:authentication:password-protected";
Date now = new Date();
```

```
request.setAttribute("oracle.security.fed.authn.engineid", TEST_ENGINE_ID);
request.setAttribute("oracle.security.fed.authn.userid", userID);
request.setAttribute("oracle.security.fed.authn.refid", refid);
request.setAttribute("oracle.security.fed.authn.authnmech", authnMethod);
request.setAttribute("oracle.security.fed.authn.authntime", now);

request.getSession().getServletContext().getContext("/fed").getRequestDispatcher("
/user/loginsso").forward(request, response);
%>
```

Since the Oracle Single Sign-On framework sets cookies in the user's browser, the authentication engine should be integrated into the logout flow; see Section 10.5, "Logout".

10.3.4 Sample Authentication Module for LDAP Integration

This section shows how to integrate a customized authentication engine with a standalone LDAP server.

Setup

The user data store configured in Fusion Middleware Control for Oracle Identity Federation references the LDAP server used by the authentication engine.

Packaging

The authentication engine consists of a Web application with a root context set to `/engine`, and contains two JSP pages:

- `loginpage.jsp`, which processes the incoming request from the federation server, and displays the login page.
- `ldapforward.jsp`, which authenticates the user's credentials against the LDAP server; upon success it forwards the user to the federation server.

Adding the Engine

To add the engine:

1. Go to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **Authentication Engines**. Select the **Authentication Engines - Custom** tab.
3. To add an authentication engine, click **Add** and enter a name for that authentication engine. Oracle Identity Federation will generate an ID for that new engine: this ID is reference by `TEST_ENGINE_ID` for this test
4. Select the authentication engine to modify it:
   - Enable the engine
   - Set "`/engine`" as the Web Context of the authentication engine
   - Set "`/loginpage.jsp`" as the Login Relative Path of the authentication engine
5. Save the changes.

See Also: Section 5.15, "Configuring Authentication Engines"

Implementation of loginpage.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
```

```
<%@page language="java" import="java.net.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

String refid = request. getAttribute("oracle.security.fed.authn.refid");

String postURL = "/engine/ldapforward.jsp?refid=" +
   (refid != null ? URLEncoder.encode(refid) : "");

String msg = request.getParameter("message");
%>
<HTML>
<BODY>
<FORM action="<%=postURL%>" method="POST">
<% if(msg != null && msg.length() > 0) { %> <%=msg%><BR/> <%}%>
Username: <INPUT type="text" name="username"/><BR/>
Password: <INPUT type="password" name="password"/><BR/>
<INPUT type="submit" value="Submit"/>
</FORM>
</BODY>
</HTML>
```

Implementation of forward.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.util.*, javax.naming.*,
javax.naming.directory.*, java.net.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

String refid = request.getParameter("refid");
String authnMethod = "oracle:fed:authentication:password-protected";

String userID = request.getParameter("username");
String password = request.getParameter("password");
Date now = new Date();

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://mynode.us.mycorp.com:389");
env.put(Context.SECURITY_AUTHENTICATION, "simple");
env.put(Context.SECURITY_PRINCIPAL, "cn=" + userID +
",cn=users,dc=us,dc=oracle,dc=com");
env.put(Context.SECURITY_CREDENTIALS, password);

try {
   DirContext ctx = new InitialDirContext(env);
} catch (NamingException ex) {
   String redirectURL = "/engine/loginpage.jsp?refid=" +
      (refid != null ? URLEncoder.encode(refid) : "") + "&message=" +
URLEncoder.encode(ex.toString() + " for " + userID);
   response.sendRedirect(redirectURL);
   return;
}

request.setAttribute("oracle.security.fed.authn.engineid", TEST_ENGINE_ID);
request.setAttribute("oracle.security.fed.authn.userid", userID);
request.setAttribute("oracle.security.fed.authn.refid", refid);
request.setAttribute("oracle.security.fed.authn.authnmech", authnMethod);
request.setAttribute("oracle.security.fed.authn.authntime", now);

request.getSession().getServletContext().getContext("/fed").getRequestDispatcher("
/user/loginsso").forward(request, response);
%>
```

Logout

Since no cookies are set in this flow, the authentication engine is not required to integrate with the logout flow (described in Section 10.5, "Logout").

10.4 Creating a Custom SP Integration Engine

This section explains how to plan, develop, and implement a custom SP integration engine.

See Also:

- Section 10.2.1, "Architecture" for a description of the SP integration engine and how it fits into Oracle Identity Federation architecture.

10.4.1 Planning a Custom SP Integration Engine

The steps for developing a custom SP integration engine involve:

- creating a service to process requests from Oracle Identity Federation in SP mode
- implementing a module to create an authenticated session for a user at the IAM server
- redirecting the user to the final target URL
- deciding whether the SP integration engine will set a cookie after it creates an authenticated session at the IAM server. If so, the engine needs to be integrated into the logout process (Section 10.5, "Logout").
- packaging these services and module into a web application, and deploying it to Oracle WebLogic Managed Server where the federation server is running
- configuring Oracle Identity Federation to reference the new SP integration module.
- if the SP integration engine accesses a user repository, ensuring that it is the same user data store configured in Fusion Middleware Control for use by Oracle Identity Federation

10.4.2 Developing and Implementing the Integration Module

This section describes how to develop the integration module and how to implement it in the federation environment.

- Path URLs
- Adding or Modifying an SP Integration Engine
- Implementing the Service

10.4.2.1 Path URLs

Communication between the Oracle Identity Federation engine and the SP integration engine requires internal servlet forwards that are equivalent to API calls. These forwards are achieved with the following JavaEE API:

```
ServletContext.getContext(String contextPath)
   .getRequestDispatcher(String relativePath)
   .forward(HttpServletRequest request,
      HttpServletResponse response)
```

where

- `contextPath` is the root context path of the web application. For example, the `contextPath` of Oracle Identity Federation is `/fed`.
- `relativePath` is the service URL to which the uses is forwarded, and is relative to the `contextPath`. For example, when starting a Federation SSO flow, the SP integration engine uses `/sp/startsso` as the relativePath when forwarding the user.

Oracle Identity Federation needs to be aware of the ID and the `contextPath` and the `relativePath` of the new SP integration engine; This is the URL that will process the result of the Federation SSO operation after the federation server has processed the incoming assertion.

10.4.2.2 Adding or Modifying an SP Integration Engine

To add or modify the SP integration engine take these steps:

1. Go to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to Administration then SP Integration Modules. Click the **Custom SP Engine** tab.
3. To add an SP integration engine, click **Add** and enter a name for that SP integration engine. Oracle Identity Federation will generate an ID for that new engine

   Note: the name will only be used in Fusion Middleware Control for display purposes while the ID is used during communication between Oracle Identity Federation and the authentication engine

4. To modify an SP integration engine, select it and:
   - Enable or disable the engine
   - Specify the `contextPath` of the SP integration engine in the Web Context field
   - Specify the relative path of the login service of the SP integration engine in the **Login Relative Path** field
   - Select the authentication mechanism to use if a local authentication procedure needs to occur during the assertion processing (this can happen when Federated Identities are used to map the assertion to a user record, if the Federation record does not exist: in this case, the user will need to be locally authenticated in order to perform the one time operation that will create the record)
   - Enable or disable logout

- Specify the relative path of the logout service of the SP integration engine in the Logout Relative Path field

5. Save your changes.

See Also: Section 5.16, "Configuring SP Integration Modules"

10.4.2.3 Implementing the Service

Upon receiving a request from Oracle Identity Federation, the SP integration engine needs to:

- create an authenticated session for the user
- redirect the user to the final URL

Attributes Set by SP Integration Module

To initiate a Federation SSO operation, the SP integration module needs to forward the user to Oracle Identity Federation by specifying the following data as `HttpServletRequest` attributes:

- An optional authentication mechanism specifying to the SP which authentication mechanism to request the IdP to use during authentication. (identified by oracle.security.fed.sp.authnmech). This parameter is ignored if usedefault is true or missing.

  Note: if set, this parameter is used to determine the IdP to use, as described below, disregarding the usedefault parameter

- An optional Provider ID referencing the IdP to use for the Federation SSO. If missing, Oracle Identity Federation will use the IdP mapped for the specified authentication mechanism. If no IdP could be found, Oracle Identity Federation will use the IdP configured as the default SSO IdP. The value is a `String` (identified by `oracle.security.fed.sp.providerid`)
- An optional federation ID referencing the affiliation to use to trigger the Federation SSO, as a String (identified by `oracle.security.fed.sp.federationid`)
- The relay state. It can contain a small string, for example a reference to some data saved in a repository or a small URL pointing to the protected resource to redirect the user to after completion of the SSO operation. (identified by `oracle.security.fed.sp.relaystate`)
- The identifier of the SP engine that started the SSO flow, as a `String` (identified by `oracle.security.fed.sp.engineid`)
- An optional `boolean` indicating if Oracle Identity Federation should authenticate the user locally using the authentication engines or if a Federation SSO should be started by redirecting the user to an IdP for authentication, as a `Boolean` (identified by `oracle.security.fed.sp.localauthn` ; default is `false`)
- A `Boolean` object indicating whether to use the configuration stored in Oracle Identity Federation or to only start the SSO based on the information being passed by the SP engine, except the IdP (identified by `oracle.security.fed.sp.usedefault`). If missing, `true` is assumed.
- A `Boolean` object indicating whether the SP should ask the IdP to challenge the user even if he/she is already authenticated (identified by

oracle.security.fed.sp.forceauthn). This parameter is ignored if usedefault is true or missing.

- A Boolean object indicating whether the SP should allow the IdP to create a federation record if one does not yet exist, during the SSO operation (identified by oracle.security.fed.sp.allowfedcreation). This parameter is ignored if usedefault is true or missing.
- A Boolean object indicating whether the SP should ask the IdP not to interact with the user during the SSO operation (identified by oracle.security.fed.sp.passive). This parameter is ignored if usedefault is true or missing.
- A String representing the binding to use when sending the AuthnRequest (identified by oracle.security.fed.sp.requestbinding). This parameter is ignored if usedefault is true or missing. Acceptable values are httpredirect, httpost, httppostsimple depending on the protocol
- A String representing the binding to use when sending the Response with the assertion (identified by oracle.security.fed.sp.responsebinding). This parameter is ignored if usedefault is true or missing. Acceptable values are artifact or httpost depending on the protocol.
- An optional authentication mechanism comparison specifying to the SP which authentication context comparison to request the IdP to use during authentication. (identified by oracle.security.fed.sp.authnmechcomparison). This parameter is ignored if usedefault is true or missing.
- A String representing the NameID format the SP will ask to the IdP for the SSO operation (identified by oracle.security.fed.sp.nameidformat). This parameter is ignored if usedefault is true or missing.
- Optional attributes to be requested from the identity provider during the Federation SSO operation (for example when interacting with an OpenID IdP). The data is passed as a Map with Strings as keys and a set of objects as values, identified by oracle.security.fed.sp.attributes. The values is optional, while the keys contain the attribute names.

Oracle Identity Federation Assertion Processing

Oracle Identity Federation then performs a SAML/WS-Fed SSO operation with a remote IdP, processes the assertion, maps it optionally to a local user record and finally forwards the user back to the SP integration engine that initiated the operation by specifying the following information as HttpServletRequest attributes:

- A Boolean object indicating if the SSO operation was successful (identified by oracle.security.fed.sp.authnresult)
- The identifier of the user as a String (identified by oracle.security.fed.sp.userid)
- Authentication time as a Date object (identified by oracle.security.fed.sp.authntime)
- Expiration time of the authenticated session as a Date (identified by oracle.security.fed.sp.expirationtime)
- The authentication mechanism used to identify the user as a String (identified by oracle.security.fed.sp.authnmech)
- The relay state as a String (identified by oracle.security.fed.sp.relaystate)
- The contents of the assertion: the NameID, the issuer of the assertion and the optional attributes. Note: the content of the assertion is not passed as XML Data,

that is the original assertion will not be passed back to the module. The data is passed as a Map with `Strings` as keys and `Set of Objects` as values (identified by `oracle.security.fed.sp.attributes`). The extra data is referenced as:

  - `orafed-nameid-value` containing the Name ID value
  - `orafed-nameid-qualifier` containing the Name ID qualifier
  - `orafed-nameid-format` containing the Name ID format
  - `orafed-providerid` containing the Peer ProviderID
  - `orafed-assertionid` - the ID of the assertion
  - `orafed-xmlmessage` - the optional XML message containing the assertion

    See Section 6.13.2, "Providing XML Message to SP Engine after SSO Completes" for details.

- The top status of the SAML Response as a String (identified by `oracle.security.fed.sp.topstatus`)
- The low status of the SAML Response if any, as a String (identified by `oracle.security.fed.sp.lowstatus`)
- The status message if any as a String (identified by `oracle.security.fed.sp.statusmessage`)
- The ProviderID that created the SSO assertion as a String (identified by `oracle.security.fed.sp.providerid`)
- The identifier of the SP engine that will process the above information (identified by `oracle.security.fed.sp.engineid`)
- A String containing the Oracle Identity Federation identifier of the user session. Oracle Identity Federation is passing the sessionID of the user session to the SP engine, so that it can persist state linked to the user, and it can reference that data by using the sessionID value. Later on, when the logout flow is being executed, Oracle Identity Federation passes the sessionID that is being logged out to the engine, so that the engine can delete the data that was used for this user session. (identified by `oracle.security.fed.sessionid`)

Authenticated Session Creation

Using this data, the SP integration engine creates an authenticated session and redirects the user to the final target URL.

If the service needs to set cookies, the cookie path must be set to "/". This is necessary because of the internal forwards between the Oracle Identity Federation and SP integration engine web applications; the user's browser needs to send the cookies related to the SP integration engine, even when accessing only the federation server. This way, when an internal forward occurs from the federation server to the SP integration engine, the cookie set by the latter is available in the HTTP Request.

10.4.3 Sample Integration Modules

The next two sections provide examples of implementing a custom authentication engine:

- Sample Integration Module 1: Oracle WebLogic Server JavaEE Integration
- Sample Integration Module 2: Customized Single Sign-On Integration

---

Note: Oracle strongly discourages users from deploying any applications on the Oracle WebLogic Managed Server other than the ones for custom integration and authentication described as sample integration modules 1 and 2 below, because doing so introduces potential security risks. Extraneous applications deployed in the Oracle WebLogic Managed Server can potentially affect the security of the federation server by allowing rogue software to change the behavior of the server flows.

10.4.4 Sample Integration Module 1: Oracle WebLogic Server JavaEE Container Integration

This section shows a simple SP integration engine that uses the `javax.servlet.http.HttpSession` to set an attribute. The presence of this attribute shows whether a user is authenticated.

Note: The example in this section is intended for illustration only and should not be used in a production environment. Indeed, it supposes that other applications deployed on the Oracle WebLogic Managed Server will consume data set by the SP integration engine, which is an approach strongly discouraged by Oracle. Furthermore, this example might not function properly in certain deployments, especially when propagating `HttpSession` across J2EE applications.

Setup

The SP integration engine will not interact with the user data store used by Oracle Identity Federation.

Packaging

The SP integration engine consists of a Web application with a root context set to `/engine`, and contains two JSP pages:

- `wlsintegration.jsp`, which processes the request from the federation server and creates an HttpSession with a `feduserid` attribute containing the user's identifier
- `application.jsp`, which serves as an application. It looks for the `HttpSession's feduserid` attribute, and triggers a Federation SSO if the attribute is not found

Adding or Modifying an SP Integration Engine

To add or modify the SP integration engine take these steps:

1. Go to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **SP Integration Modules**. Click the **Custom SP Engine** tab.
3. To add an SP integration engine, click **Add** and enter a name for that SP integration engine. Oracle Identity Federation generates an ID for that new engine: this ID is reference by `TEST_ENGINE_ID` for this test
4. To modify an SP integration engine, select it and:
   - Enable the engine
   - Set "`/engine`" as the Web Context of the authentication engine

- Set "`/wlsintegration.jsp`" as the Login Relative Path of the SP integration engine
- Select the authentication mechanism to use if a local authentication procedure needs to occur during the assertion processing (this can happen when Federated Identities are used to map the assertion to a user record, if the Federation record does not exist: in this case, the user will need to be locally authenticated in order to perform the one time operation that will create the record)

5. Save your changes.

See Also: Section 5.16, "Configuring SP Integration Modules"

Implementation of application.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.net.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

String userid = (String)request.getSession().getAttribute("feduserid");
if (userid == null || userid.length() == 0) {
           request.setAttribute("oracle.security.fed.sp.engineid", TEST_ENGINE_
ID);
request.setAttribute("oracle.security.fed.sp.usedefault", Boolean.TRUE);
request.setAttribute("oracle.security.fed.sp.relaystate",
"/engine/application.jsp");

request.getSession().getServletContext().getContext("/fed").getRequestDispatcher("
/sp/startsso").forward(request, response);
 return;
}
%>
Welcome <%=userid%>
```

Implementation of wlsintegration.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.util.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

String userid = (String)request.getAttribute("oracle.security.fed.sp.userid");
String targetURL  =
(String)request.getAttribute("oracle.security.fed.sp.relaystate");

request.getSession().setAttribute("feduserid", userid);
response.sendRedirect(targetURL);
%>
```

Logout

Since this application sets up an HttpSession in the Oracle WebLogic Managed Server instance, the SP integration engine must be integrated in the logout flow (see

Section 10.5, "Logout").

10.4.5 Sample Integration Module 2: Customized Single Sign-On Integration

This section shows an SP integration engine that uses a simple Single Sign-On framework based on a cookie containing the username and the expiration time of the authenticated session.

Note: This example is intended for illustration only and should not be used in a production environment. For example, the cookies set in this example are not encrypted, allowing an attacker to impersonate a user by manually constructing such cookies.

Setup

The SP integration engine will not interact with the user data store used by Oracle Identity Federation. The engine will set up a cookie, for the entire domain, containing the user's identifier as a `String` variable and the session timeout as a long.

Packaging

The SP integration engine consists of a Web application with a root context set to `/engine`, and contains two JSP pages:

- `domainintegration.jsp`, which processes the request from the Oracle Identity Federation server and creates a cookie with the user ID and session timeout
- `domainapplication.jsp`, which serves as an application. It looks for the cookie and triggers a federation SSO if the cookie is not found.

Adding or Modifying an SP Integration Engine

To add or modify the SP integration engine take these steps:

1. Go to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **SP Integration Modules**. Click the **Custom SP Engine** tab.
3. To add an SP integration engine, click **Add** and enter a name for that SP integration engine. Oracle Identity Federation will generate an ID for that new engine: this ID is reference by **TEST_ENGINE_ID** for this test
4. To modify an SP integration engine, select it and:
   - Enable the engine
   - Set "**/engine**" as the Web Context of the authentication engine
   - Set "**/domainintegration.jsp**" as the Login Relative Path of the SP integration engine
   - Select the authentication mechanism to use if a local authentication procedure needs to occur during the assertion processing (this can happen when Federated Identities are used to map the assertion to a user record, if the Federation record does not exist: in this case, the user will need to be locally authenticated in order to perform the one-time operation that will create the record)
5. Save your changes.

See Also: Section 5.16, "Configuring SP Integration Modules"

Implementation of domainapplication.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.net.*, java.util.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

Cookie[] cookies = request.getCookies();
String userid = null;
Date timeout = null;
for(int i = 0, size = (cookies != null ? cookies.length : 0); i < size; i++) {
   String name = cookies[i].getName();
   if ("spintegrationcookie".equals(name)){
      String value = cookies[i].getValue();
      StringTokenizer st = new StringTokenizer(value, "*");
      userid = st.nextToken();
      timeout = new Date(Long.parseLong(st.nextToken()));
      break;
   }
}
if (userid == null || userid.length() == 0) {
request.setAttribute("oracle.security.fed.sp.engineid", TEST_ENGINE_ID);
request.setAttribute("oracle.security.fed.sp.usedefault", Boolean.TRUE);
request.setAttribute("oracle.security.fed.sp.relaystate",
"/engine/domainapplication.jsp");
request.getSession().getServletContext().getContext("/fed").getRequestDispatcher("
/sp/startsso").forward(request, response);
   return;
}
%>
Welcome <%=userid%>. You are logged until <%=timeout%>
```

Implementation of domainintegration.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.util.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

String userid = (String)request.getAttribute("oracle.security.fed.sp.userid");
String targetURL  =
(String)request.getAttribute("oracle.security.fed.sp.relaystate");
Date expirationInst =
(Date)request.getAttribute("oracle.security.fed.sp.expirationtime");
String cookieValue = userid + "*" + expirationInst.getTime();
Cookie cookie = new Cookie("spintegrationcookie", cookieValue);
cookie.setDomain(".us.example.com");
cookie.setPath("/");
response.addCookie(cookie);
response.sendRedirect(targetURL);
%>
```

Logout

Since this sample application sets up a domain cookie, the SP integration engine must be integrated into the logout flow (see Section 10.5, "Logout").

10.5 Logout

This section explains how to configure logout flows.

10.5.1 Changing Logout Flow

This section contains topics relevant to redirection during logout.

URLs

During the logout operations, the user is being redirected between the federation engine and the logout service of the authentication and SP integration engines.

Oracle Identity Federation needs to be aware of the location of the logout service in order to redirect the user to the servlet/jsp page for logout. This URL is defined in the logout location field of the authentication and SP integration engines. The URL can be defined as the union of the Web Context of the engine and the logout relative path.

Implementing the Logout Service

The operations that need to be performed by the logout service include:

- processing requests from the federation engine, logging the user out of the IAM framework and sending the user back to Oracle Identity Federation
- processing requests from the IAM framework, sending the user to Oracle Identity Federation for logout, receiving the user back from Oracle Identity Federation after logout and sending the user to the IAM framework

Oracle Identity Federation invokes Authn/SP Engine

When Oracle Identity Federation sends the user to the authentication/SP engine, it will:

1. Perform an internal forward to the web context and relative logout path of the engine
2. Specify the engine ID of the invoked engine on the HttpServletRequest the attribute referenced by `oracle.security.fed.authn.engineid` (if the engine is an authentication engine) or oracle.security.fed.sp.engineid (if the engine is an SP engine)
3. Optionally specify on the `HttpServletRequest` the attribute referenced by `oracle.security.fed.sessionid` that will hold the identifier of the user session being logged out (`String` object)

When the authentication/SP engine logs the user out, and internally forwards the user back to Oracle Identity Federation, it:

1. Performs an internal forward to the `/fed` web context and `/user/logoutretsso`
2. Specifies the engine ID of the invoked engine on the `HttpServletRequest` the attribute referenced by `oracle.security.fed.authn.engineid` (if the engine is an authentication engine) or `oracle.security.fed.sp.engineid` (if the engine is an SP engine), and that attribute value is the identifier of the engine that performed the logout operation

Authn/SP invokes Oracle Identity Federation

When an authentication/SP engine invokes Oracle Identity Federation for logout, it:

1. Performs an internal forward to the `/fed` web context and `/user/logoutsso`
2. Specifies the engine ID of the invoked engine on the `HttpServletRequest` the attribute referenced by `oracle.security.fed.authn.engineid` (if the engine is an authentication engine) or `oracle.security.fed.sp.engineid` (if the engine is an SP engine), and that attribute value is the identifier of the engine being invoked for logout
3. Specifies a return URL where Oracle Identity Federation redirects the user after logout. That URL is specified via `HttpServletRequest` attribute, referenced by `oracle.security.fed.logout.returnurl.`

At the end of the logout flow, the user is logged out from Oracle Identity Federation and redirected to the return URL.

10.5.2 Sample Logout Services

In the next two sections, these scenarios of logout services are outlined:

- Logout Service Example #1 describes a custom logout service when both the authentication and SP integration engines are customized
- Logout Service Example #2 describes a custom logout service when only the SP integration engine is customized

10.5.3 Logout Service Example #1

This section describes how to integrate a custom logout service, assuming that both the authentication and SP integration engines have been customized, that is, the default engines are not used anymore.

Setup

In this example, the authentication engine is the LDAP engine described in Section 10.3, "Creating a Custom Authentication Engine", and the SP integration engine is the Oracle WebLogic Server integration engine described in Section 10.4.4, "Sample Integration Module 1: Oracle WebLogic Server JavaEE Container Integration".

Packaging

The logout service consists of a JSP page bundled with the authentication and SP integration engines:

- `logout.jsp`, which processes the request from Oracle Identity Federation, remove the `feduserid` attribute from the HttpSession object, set in the `wlsintegration.jsp` page, and redirect the user to either Oracle Identity Federation or the `doneURL` parameter.

Updating the Engine

To update the engine:

1. Go to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **SP Integration Modules**. Click the **Custom SP Engine** tab.
3. To modify an SP integration engine, select it and:

- Enable logout engine
- Set "/logout.jsp" as the **Logout Relative Path** of the SP integration engine

4. Save your changes.

See Also: Section 5.16, "Configuring SP Integration Modules"

Implementation of logout.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.net.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

request.getSession().removeAttribute("feduserid");

request.setAttribute("oracle.security.fed.sp.engineid", TEST_ENGINE_ID);

request.getSession().getServletContext().getContext("/fed").getRequestDispatcher("
/user/logoutretsso").forward(request, response);
%>
```

10.5.4 Logout Service Example #2

This section describes how to integrate a custom logout service, assuming that the SP integration engine has been customized.

Setup

In this example, the SP integration engine is the customized SSO integration engine described in Section 10.4.5, "Sample Integration Module 2: Customized Single Sign-On Integration".

Packaging

The logout service consists of a JSP page bundled with the authentication and SP integration engines:

- `domainlogout.jsp`, which processes the request from Oracle Identity Federation, removes the cookie, and redirects the user to the `/logoutretsso` URL.

Updating the Engine

To update the engine:

1. Go to Fusion Middleware Control and navigate to the Oracle Identity Federation instance.
2. Navigate to **Administration**, then **SP Integration Modules** .
3. To modify the SP integration engine, select it and:
   - Enable logout engine
   - Set "`/domainlogout.jsp`" as the **Logout Relative Path** of the SP integration engine
4. Save your changes.

See Also: Section 5.16, "Configuring SP Integration Modules"

Implementation of domainlogout.jsp

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.net.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

Cookie cookie = new Cookie("spintegrationcookie", "");
cookie.setDomain(".us.example.com");
cookie.setPath("/");
cookie.setMaxAge(0);
response.addCookie(cookie);

request.setAttribute("oracle.security.fed.sp.engineid", TEST_ENGINE_ID);

request.getSession().getServletContext().getContext("/fed").getRequestDispatcher("
/user/logoutretsso").forward(request, response);
%>
```

11

Configuring Oracle Identity Federation for the Business Processing Plug-in

Oracle Identity Federation provides a plug-in framework to customize the business processing of the operations performed by the server. This chapter explains plug-in features and provides an example:

11.1 About the Business Processing Plug-in

This section describes some key facts about the plug-in framework.

11.1.1 Basic Flow of Business Processing Plug-in

The processing flow is as follows:

- You implement a plug-in that will be invoked in various sections of the business flows.
- The plug-in can analyze data collected during the execution of the operation, and decide whether an extra business step should be required.
- If any additional actions are to be performed, the plug-in returns to Oracle Identity Federation a URL where the user needs to be redirected.
- The redirection URL can contain query string parameters set by the plug-in.
- Oracle Identity Federation appends one query string parameter, referenced by `refID`, to be sent when the user is returning to Oracle Identity Federation
- Once the extra operation is performed, the user must be redirected to Oracle Identity Federation with the `refid` parameter, to the following URL:

```
http(s)://OIF-HOST:OIF-PORT/fed/user?refid=VALUE_RETRIEVED_FROM_REDIRECT_URL
```

11.1.2 Implementation

The tasks needed to implement the business processing plug-in are:

- Build the plug-in.

  See Section 11.1.3, "Building the Plug-in, Operations and Parameters".
- Register the plug-in in the Oracle Identity Federation configuration file.

  See Section 11.2, "Configuring the Business Processing Plug-in".
- Package the plug-in in a JAR file and add this file to the CLASSPATH.

  See Section 11.3, "Packaging the Plug-in".
- Set up JavaEE security.

  See Section 11.4, "Configuring JavaEE Security"
- Restart Oracle WebLogic Server.

11.1.3 Building the Plug-in, Operations and Parameters

Building the Plug-in

The plug-in will need to extend the `oracle.security.fed.plugins.bizops.OperationListener` interface, and will need to implement the "`public ListenerResult process(int operationType, OperationData params)`" method.

This method has two arguments; the first is the type of operation being performed and the second includes parameters related to the operation that allow the plug-in to make a decision. The method returns a `ListenerResult` class containing a status and an optional `redirectURL`. If the status is OK, Oracle Identity Federation resumes its operations, otherwise it redirects the user to the specified redirection URL.

Operations

The operations include:

- OperationTypes.BUSINESS_IDP_CREATE_PERSISTENT_FEDERATION: indicates a persistent federation is created on the IdP side
- OperationTypes.BUSINESS_IDP_CREATE_TRANSIENT_FEDERATION: indicates a transient federation is created on the IdP side
- OperationTypes.BUSINESS_IDP_SSO: indicates an SSO operation performed on the IdP side

Parameters Passed

The parameters passed in the `OperationData` object are:

- BusinessProcessingConstants.DATA_STRING_PROVIDERID: references the Service Provider ID. Type is `String`
- BusinessProcessingConstants.DATA_STRING_USERID: references the User ID. Type is `String`
- BusinessProcessingConstants.DATA_STRING_SESSIONID: references the Session ID. Type is `String`

- BusinessProcessingConstants.DATA_STRING_NAMEID_FORMAT: references the Name ID Format of the federation being created. Type is `String`
- BusinessProcessingConstants.DATA_STRING_PROTOCOL_VERSION: references the protocol being executed. Type is `String`
- BusinessProcessingConstants.DATA_BOOLEAN_AUTHNREQUEST_ISPASSIVE: references the `IsPassive` field from the `AuthnRequest`. Type is `Boolean`

The returned status values of the `ListenerResult` class can be:

- BusinessProcessingConstants.STATUS_OK: indicates that the plug-in does not require any particular action.
- BusinessProcessingConstants.STATUS_REDIRECT: indicates that the plug-in wishes to redirect the user to a URL.

11.2 Configuring the Business Processing Plug-in

Follow these steps to add a plug-in to the Oracle Identity Federation configuration file:

1. Open the `$DOMAIN_HOME/config/fmwconfig/servers/wls_oif1/applications/OIF_11.1.1.2.0/configuration/config.xml` file
2. Locate the `Config` XML element whose attribute name is `serverconfig`.
3. Locate the `PropertiesList` XML element whose attribute name is `businessprocessingplugins`.
4. Add a `Property` XML child element to the `PropertiesList`. The text child of the `Property` element should be the classname of the plug-in, and the type attribute of this element should be `string`.
5. Save and exit.

Here is an example of the configured file:

```
<FederationConfig xmlns="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd"
version="0" activationenabled="false">
   <Config name="serverconfig">
        ...
       <PropertiesList name="businessprocessingplugins">
          <Property
type="string">oracle.security.fed.plugins.BusinessProcessingSample</Property>
       </PropertiesList>
        ...
   </Config>
   ...
</FederationConfig>
```

11.3 Packaging the Plug-in

Take these steps to package your plug-in:

1. Add the plug-in to a jar file.
2. Copy the jar file to the Oracle WebLogic Server `lib` directory:

   ```
   Oracle/Middleware/user_projects/domains/IDMDomain/lib
   ```

3. Copy other required jar files to the same directory:

- Copy `oif.jar` from `Oracle/Middleware/Oracle_IDM1/fed/jlib/` to `Oracle/Middleware/user_projects/domains/IDMDomain/lib`.

Note: Repeat this step every time after you apply a patch set.

- Copy `commons-httpclient-3.1.jar` to the same directory.
- Copy `commons-codec-1.2.jar` to the same directory.

For details about the environment configuration, see Setting Up Environment Variables in the *Oracle Fusion Middleware Administrator's Guide*.

11.4 Configuring JavaEE Security

Update the WebLogic policy file which resides in this location:

```
Oracle/Middleware/wlserver_10.3/server/lib/weblogic.policy
```

Add these lines to the file:

```
grant codeBase "file:${user.domain}/lib/-" {
  permission java.security.AllPermission;
  };
  grant codeBase
  "file:/home/oracle/Oracle/Middleware/user_projects/domains/IDMDomain/lib/-"
  {
  permission java.security.AllPermission;
  };
```

11.5 Example of Plug-in and Redirect Page

A sample plug-in might look like this:

```
package oracle.security.fed.plugins;

import java.net.URLEncoder;
import java.util.Set;
import java.util.HashSet;

import oracle.security.fed.plugins.bizops.BusinessProcessingConstants;
import oracle.security.fed.plugins.bizops.BusinessProcessingException;
import oracle.security.fed.plugins.bizops.ListenerResult;
import oracle.security.fed.plugins.bizops.OperationData;
import oracle.security.fed.plugins.bizops.OperationListener;
import oracle.security.fed.plugins.bizops.OperationTypes;

// in this example, the plug-in will redirect the user to an external page the
first time a user
// creates a persistent federation. Later on, if the user creates another
federation (with the same
// provider or another one), the plug-in will not redirect the user anymore.
// Note: restarting the server will wipe out the cached information from the
plug-in, resetting the data
// indicating whether or not any user was already redirected to the external page.

public class BusinessProcessingSample implements OperationListener {

    private Set licenseAgreements = new HashSet();
    public ListenerResult process(int operationType, OperationData params)
        throws BusinessProcessingException {
        ListenerResult result = new
ListenerResult(BusinessProcessingConstants.STATUS_OK);

        switch(operationType)
        {
            case OperationTypes.BUSINESS_IDP_CREATE_PERSISTENT_FEDERATION:
                  String userid =
params.getStringProperty(BusinessProcessingConstants.DATA_STRING_USERID);
                  if (!licenseAgreements.contains(userid))
                  {
                      // redirect to remote page
                      result.setStatus(BusinessProcessingConstants.STATUS_
REDIRECT);

                      StringBuffer sb = new StringBuffer();

sb.append("http://WEB-SERVER-HOST:WEB-SERVER-PORT/businesstest.jsp?providerid=");

sb.append(URLEncoder.encode(params.getStringProperty(BusinessProcessingConstants.D
ATA_STRING_PROVIDERID)));
                      sb.append("&userid=");

sb.append(URLEncoder.encode(params.getStringProperty(BusinessProcessingConstants.D
ATA_STRING_USERID)));
                      result.setRedirectURL(sb.toString());

                      // add the user to the license agreement set
                      licenseAgreements.add(userid);
                  }
                  break;
        }

        return result;
    }
}
```

Here is a sample redirect page:

```
<%@ page language="java"
    import="java.net.*"%>
<%
// Set the Expires and Cache Control Headers
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

String providerid = request.getParameter("providerid");
String userid = request.getParameter("userid");
String refid = request.getParameter("refid");

String returnurl = "http://OIF-HOST:OIF-PORT/fed/user?refid=" +
URLEncoder.encode(refid);
%>

<html>
<body>
License Agreeement approved for:
ProviderID = <%=providerid%>
```

```
<BR>
UserID = <%=userid%>
<BR>
<a href="<%=returnurl%>">Click here to resume flow</a>

</body>
</html>
```

11.6 Business Processing Plug-in API

The Business Processing Plug-in API (javadoc) is available at:

Oracle Fusion Middleware Business Processing Plug-in Java API Reference for Oracle Identity Federation

12 Implementing Custom Actions

Custom actions enable site-specific operations to be executed during an Oracle Identity Federation flow before and after an authentication engine or SP integration engine is invoked.

Oracle Identity Federation implements custom actions using the pre- and post-processing action plug-ins. This chapter explains how to implement custom actions.

Note: Oracle Identity Federation is also referred to here as the federation server.

12.1 Introduction to Custom Actions

The pre- and post-processing plug-ins are implemented as JSP or JavaEE servlets, invoked during an authentication flow:

- before and after invoking an authentication engine
- before and after invoking an SP integration engine

You can use custom actions with all authentication engines and SP integration engines, including custom engines. This section explains how the actions work and how they interact with Oracle Identity Federation:

12.1.1 Pre- and Post-Processing Custom Actions for Authentication Engines

Oracle Identity Federation invokes an authentication engine to identify the user when required at run-time. The following steps describe the typical flow of events.

1. Oracle Identity Federation takes these steps:
   - determines that the user must be identified

- selects the authentication engine to be used
- retrieves the location of the authentication engine
- internally forwards the user to the authentication engine, specifying the root context and the relative path of the engine's endpoint

2. The authentication engine, which is a pluggable module that interacts with external authentication providers to establish user identity, takes these steps:
   - interacts with the user to obtain credentials. For example, an LDAP authentication engine displays a login page to accept user credentials, which are then validated against an LDAP directory.
   - if successful, internally forwards the user back to Oracle Identity Federation with the authentication information, which consists of the user identifier and the time at which identity was established.
3. Oracle Identity Federation analyzes the information and creates or updates the user session.

You can use custom actions to:

- manipulate the data exchanged between the federation server and the authentication engine; for example, to construct email address from the username johndoe->johndoe@mycompany.com
- perform additional steps during authentication; for example, to contact an external data source or system to obtain more information about the user.

To set up a custom action plug-in:

- implement a pre-processing action plug-in to be performed before invoking the authentication engine
- implement a post-processing plug-in for any actions or changes to be performed after authentication, when the user is redirected from the authentication engine to Oracle Identity Federation
- deploy the plug-in to the WebLogic managed server where Oracle Identity Federation is running
- configure Oracle Identity Federation to invoke the plug-in instead of the authentication engine so that the plug-in can perform pre-processing tasks
- configure the authentication engine to invoke the plug-in instead of redirecting to Oracle Identity Federation, so that the plug-in can perform post-processing tasks.

Note: If the default (out-of-the-box) authentication engine is in use, you will need to modify the Oracle Identity Federation configuration; if a custom engine is in use, you will need to update the engine.

12.1.2 Pre- and Post-Processing Custom Actions for SP Integration Engines

When Oracle Identity Federation is acting as a service provider, either:

- the Identity and Access Management (IAM) system, for example Oracle Access Manager, invokes Oracle Identity Federation to perform a federated SSO operation to authenticate the user at a remote identity provider, or
- a remote identity provider initiates a federated SSO operation for the user to access a protected resource in the SP domain

When the IAM system invokes Oracle Identity Federation for federated SSO, the flow is as follows:

1. A user attempts to access a resource protected by the IAM system.
2. The IAM system determines that the user needs to be authenticated by means of a federated SSO operation.
3. The IAM system redirects the user to the Oracle Identity Federation SP engine with which it is integrated.
4. The Oracle Identity Federation SP integration module, which could be a custom module, performs some operations and internally forwards the user to Oracle Identity Federation, specifying the information needed for the operation. For example, it can specify the authentication mechanism to use, the relay state, and so on.
5. Oracle Identity Federation processes the information, triggers the federated SSO operation, and redirects the user to the remote identity provider for authentication.
6. The identity provider identifies the user, creates an assertion, and redirects the user back to Oracle Identity Federation/SP.
7. Oracle Identity Federation validates the assertion, maps it to a user in the local domain, and creates an Oracle Identity Federation session for the user.
8. Oracle Identity Federation internally forwards the user back to the SP integration module that triggered the flow (or to the default SP integration module in case of an IdP-initiated SSO operation).
9. The SP integration module processes the data, creates a web access session, and redirects the user to the protected resource.
10. The IAM system grants the user access to the resource.

You can use custom actions to customize the data exchanged between the federation server and the SP integration module, and perform certain actions in the process.

To set up a custom action plug-in:

– implement a pre-processing plug-in when actions or changes must occur before the SP Integration Module redirects the user to Oracle Identity Federation to start the flow.

– implement a post-processing plug-in when actions or changes must happen after the federated SSO operation, when the user is redirected from Oracle Identity Federation to the SP Integration Module.

– deploy the plug-in to the WebLogic Managed Server where Oracle Identity Federation is running

– configure the SP integration module to invoke the plug-in instead of redirecting to Oracle Identity Federation, so that the plug-in can perform the custom tasks.

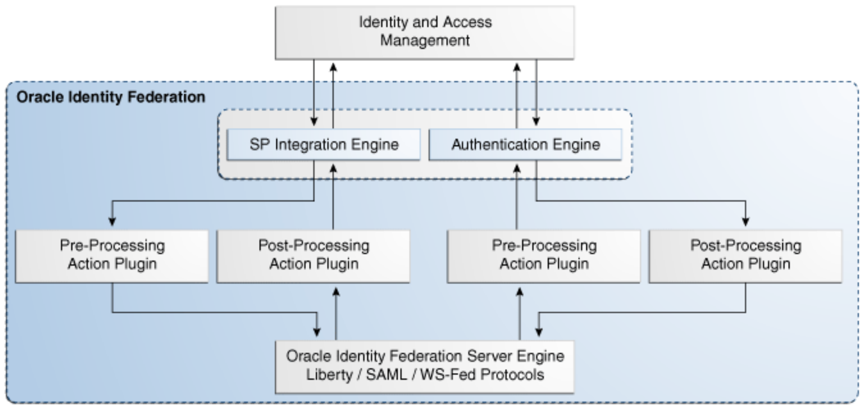
Note: If the default (out-of-the-box) SP integration module is in use, you will need to modify the Oracle Identity Federation configuration; if a custom engine is in use, you will need to update the engine.

- configure Oracle Identity Federation to invoke the plug-in instead of the SP integration module, so the plug-in can perform the custom tasks.

12.1.3 Custom Actions Architecture

Figure 12–1 explains the custom actions plug-in architecture:

Figure 12–1 Custom Action Plug-ins


Identity and Access Management
Oracle Identity Federation
SP Integration Engine
Authentication Engine
Pre-Processing Action Plugin
Post-Processing Action Plugin
Pre-Processing Action Plugin
Post-Processing Action Plugin
Oracle Identity Federation Server Engine Liberty / SAML / WS-Fed Protocols

In this figure, Oracle Identity Federation is customized and configured to invoke plug-ins:

- before the SP integration engine invokes Oracle Identity Federation
- before Oracle Identity Federation invokes the SP integration engine
- before Oracle Identity Federation invokes the authentication engine
- before the authentication engine invokes Oracle Identity Federation

12.1.3.1 Flow for Oracle Identity Federation as SP

During a federated SSO operation where Oracle Identity Federation acts as the service provider, the flow is as follows:

1. An Identity and Access Management (IAM) module such as Oracle Access Manager invokes the SP integration engine to start a federated SSO operation. The SP integration engine invokes the pre-processing plug-in for the SP engine to perform custom actions.
2. The pre-processing plug-in for SP engine invokes Oracle Identity Federation to start the federated SSO flow.
3. Oracle Identity Federation redirects the user to an IdP where the user is authenticated and an assertion created.
4. The IdP redirects the user back to Oracle Identity Federation with an assertion that is validated and mapped to a user.
5. Oracle Identity Federation bundles the user and assertion data, and invokes the post-processing plug-in for the SP engine to perform some custom tasks.
6. The post-processing plug-in invokes the SP integration engine by providing the user and assertion data.

7. The SP engine creates a use session in the IAM domain, and redirects the user to the protected resource.

12.1.3.2 Flow for Oracle Identity Federation Authenticating User

When Oracle Identity Federation needs to authenticate a user, the flow is as follows:

1. Oracle Identity Federation, as part of a runtime flow, determines that it needs to locally authenticate the user. It invokes the pre-processing plug-in for the authentication engine to perform some custom tasks.
2. The pre-processing plug-in invokes the authentication engine.
3. The authentication engine uses the IAM domain to challenge and identify the user.
4. The authentication engine bundles the authentication data and invokes the post-processing plug-in for the authentication engine to perform some custom tasks.
5. The post-processing plug-in for authentication engine invokes Oracle Identity Federation, providing the authentication data.
6. Oracle Identity Federation resumes operations.

12.2 Pre-processing Custom Action for Authentication Engine

The pre-processing plug-in is a module to which the user is directed, as part of an authentication operation, before invoking the authentication engine. The plug-in enables custom actions to be taken before authentication.

When the plug-in is in use, Oracle Identity Federation does not redirect the user to the authentication engine; rather, it forwards the user internally to the plug-in, passing it certain data for use during authentication. After performing its custom actions, the plug-in forwards the user to the correct authentication engine, along with the data originally provided by the federation server, to resume the authentication flow.

12.2.1 Implementing the Pre-processing Custom Action

Custom Action Interaction with Oracle Identity Federation

When Oracle Identity Federation redirects a user to the authentication engine, it passes certain data to the engine as attributes on the `HttpServletRequest` object. The same data is made available to pre-processing plug-ins:

- the authentication mechanism to use when challenging the user for identification (String, identified by oracle.security.fed.authn.authnmech)
- an identifier referencing the action being performed (String, identified by oracle.security.fed.authn.refid)
- the ProviderID and the description of the remote service provider for which this local authentication is requested, if a federated SSO operation is performed (String, identified by oracle.security.fed.authn.providerid and oracle.security.fed.authn.providerdescription respectively)
- the identifier referencing the engine used to authenticate the user (String, identified by oracle.security.fed.authn.engineid)
- the identifier of the user, if set (String, identified by oracle.security.fed.authn.userid)

- the Force Authentication flag, indicating whether the engine should challenge the user even if the user is already authenticated. If missing, `false` is assumed. (Boolean, identified by oracle.security.fed.authn.forceauthn)
- the Is Passive flag, indicating whether the engine is allowed to visually interact with the user. If missing, `false` is assumed (Boolean, identified by oracle.security.fed.authn.passive)
- Optionally, a map of attributes that need to be set by the engine; these attributes are required so that Oracle Identity Federation/IdP can create the assertion with the `AttributeStatement`, as specified by the configuration for that specific remote provider. (identified by oracle.security.fed.authn.attributes)

  When Oracle Identity Federation receives and processes an SSO assertion, but is not able to map the assertion to a local user, the server requests that the user be locally authenticated. The map then contains this data from the assertion:

  – orafed-nameid-value – the user's Name ID value
  – orafed-nameid-qualifier – the user's Name ID qualifier
  – orafed-nameid-format – the user's Name ID format
  – orafed-providerid – the IdP's ProviderID
  – orafed-assertionid - the ID of the assertion
  – orafed-xmlmessage - the optional XML message containing the assertion
- Optionally, the Oracle Identity Federation session identifier, if the user already has an active session. Oracle Identity Federation passes the session identifier of the existing user session to the authentication engine, so that the engine can persist the state linked to the user, and refer to that data using the sessionID value. (String, identified by oracle.security.fed.sessionid)

  Later, when the logout flow is executed, Oracle Identity Federation passes the session identifier to the engine, so that the engine can delete the data used for this user session.

See Also: Section 10.3.2, "Developing and Implementing the Authentication Module".

Forwarding the User to the Authentication Engine

After processing, the pre-processing plug-in must forward the user to the correct authentication engine (the engine that Oracle Identity Federation would invoke in the absence of the plug-in).

You can use the oracle.security.fed.authn.engineid attribute present in the `HttpServletRequest` object to determine the path to which the user must be internally forwarded. The possible engine ID values are shown in Table 12–1:

Table 12–1 Engine IDs for Authentication Engines

Engine ID	Meaning
osso	Invoke the Oracle Single Sign-On engine by performing a forward to the endpoint referenced by `/fed` as the root context and `/user/authnosso` as the relative path
oam	Invoke the Oracle Access Manager engine by performing a forward to the endpoint referenced by `/fed` as the root context and `/user/authnoam` as the relative path
ldap	Invoke the LDAP engine by performing a forward to the endpoint referenced by `/fed` as the root context and `/user/authnldap` as the relative path

Table 12–1 (Cont.) Engine IDs for Authentication Engines

Engine ID	Meaning
rdbmssec	Invoke the RDBMS security engine by performing a forward to the endpoint referenced by `/fed` as the root context and `/user/authnrdbmssec` as the relative path
rdbmstable	Invoke the RDBMS table engine by performing a forward to the endpoint referenced by `/fed` as the root context and `/user/authnrdbmstb` as the relative path
jaas	Invoke the JAAS engine by performing a forward to the endpoint referenced by `/fed` as the root context and `/user/authnjaas` as the relative path
infocard	Invoke the Infocard engine by performing a forward to the endpoint referenced by `/fed` as the root context and `/user/authnic` as the relative path
proxy	Invoke the Fed SSO Proxy engine by performing a forward to the endpoint referenced by `/fed` as the root context and `/user/authnproxy` as the relative path
httpheader	Invoke the HTTP Header engine by performing a forward to the endpoint referenced by `/fed` as the root context and `/user/authnhttp` as the relative path

Notes:

- The plug-in must provide the authentication engine with the data that was passed to the plug-in as part of the authentication flow (namely, the attributes that were set on the `HttpServletRequest` object).
- The plug-in can modify all attributes that were set on the `HttpServletRequest` object except for the oracle.security.fed.authn.authnmech, oracle.security.fed.authn.refid and oracle.security.fed.authn.engineid attributes.

12.2.2 Configuring Oracle Identity Federation for the Custom Action

Configure Oracle Identity Federation to forward the user to a pre-processing plug-in by performing these tasks:

- identify the authentication engine whose flow will be modified. Choices are: Oracle SSO, OAM, LDAP, RDBMS Security, RDBMS Table, Proxy, JAAS and Infocard engines
- Create or set the following two properties:
  – the web context property, referencing the web context where the custom JSP Page or servlet of the pre-processing plug-in resides. This property is located in the authnengines group of the Oracle Identity Federation configuration.
  – the relative path property referencing the path in the web context where the custom plug-in resides.

Table 12–2 shows how to set the properties for each authentication engine:

Table 12–2 Authentication Engine Configuration for Pre-processing Plug-in

Authentication Engine	web context property	relative path property
Oracle Single Sign-On	osso-login-context (default missing)	osso-login (default /user/authnosso)
Oracle Access Manager 10*g*	oam-login-context (default missing)	oam-login (default /user/authnoam)
Oracle Access Manager 11*g*	oam11g-login-context (default missing)	oam11g-login (default /user/spretoam11g)
LDAP	ldap-login-context (default missing)	ldap-login (default /user/authnldap)
RDBMS Security	rdbmssec-login-context (default missing)	rdbmssec-login (default /user/authnrdbmssec)
RDBMS Table	rdbmstable-login-context (default missing)	rdbmstable-login (default /user/authnrdbmstb)
JAAS	jaas-login-context (default missing)	jaas-login (default /user/authnjaas)
Infocard	infocard-login-context (default missing)	infocard-login (default /user/authnic)
Fed SSO Proxy	proxy-login-context (default missing)	proxy-login (default /user/authnproxy)
HTTP Header	httpheader-login-context (default missing)	httpheader-login (default /user/authnhttp)

Use the WLST commands to set these properties in the Oracle Identity Federation configuration.

For example, the following commands, in the WLST script environment for the Oracle Identity Federation instance, configure a pre-processing plug-in to be invoked before the Oracle SSO engine:

```
setConfigProperty('authnengines', 'osso-login-context', '/rootcontext', 'string')

setConfigProperty('authnengines', 'osso-login', '/relativepath', 'string')
```

12.3 Post-processing Custom Action for Authentication Engine

The user is directed to the post-processing plug-in module, as part of an authentication operation, after the authentication engine has completed processing and before the user is directed to Oracle Identity Federation. The plug-in enables custom actions to be taken after authentication.

When the plug-in is in use, the authentication engine forwards the user internally to the plug-in, passing it the authentication data. After performing its custom actions, the plug-in forwards the user to Oracle Identity Federation, supplying the authentication data.

Note: The available authentication engines are: Oracle SSO, OAM, LDAP, RDBMS Security, RDBMS Table, Proxy, JAAS Infocard, and HTTP Header.

12.3.1 Implementing the Post-processing Plug-in

Custom Action Interaction with Oracle Identity Federation

When the authentication engine redirects the user to Oracle Identity Federation during the authentication flow, it provides the following data to the plug-in as attributes on the `HttpServletRequest` object:

- The identifier of the user (String, identified by oracle.security.fed.authn.userid)
- Authentication time (Date, identified by oracle.security.fed.authn.authntime)
- Expiration time of the authenticated session (Date, identified by oracle.security.fed.authn.expirationtime)
- The authentication mechanism used to identify the user (String, identified by oracle.security.fed.authn.authnmech)
- The identifier referencing the action that was being performed, from the request (String, identified by oracle.security.fed.authn.refid)
- The identifier referencing the engine used to authenticate the user (String, identified by oracle.security.fed.authn.engineid)
- Optionally, a map of attributes to be stored in the user session. This map has String objects as keys and a set of objects as values (identified by oracle.security.fed.authn.attributes).
- Optionally, the Oracle Identity Federation session identifier that Oracle Identity Federation uses to reference the Oracle Identity Federation user session. This allows the engine and Oracle Identity Federation to share the same identifier to reference the user session. (String, identified by oracle.security.fed.sessionid)

  Later, when the logout flow is executed, Oracle Identity Federation passes the sessionID that is being logged out to the engine, so that the engine can delete the data used for this user session.

Notes:

- The plug-in must provide Oracle Identity Federation server with the data that was passed to it as part of the authentication flow; this consists of attributes that were set on the `HttpServletRequest` object.
- The plug-in can modify all attributes that were set on the `HttpServletRequest` object except the oracle.security.fed.authn.authnmech, oracle.security.fed.authn.refid, and oracle.security.fed.authn.engineid attributes.

12.3.2 Configuring Oracle Identity Federation for the Plug-in

To configure Oracle Identity Federation to forward the user to a post-processing plug-in following the authentication flow, set the following properties:

- Create or set the web context property, referencing the web context where the custom JSP Page or servlet resides. This property is located in the `serverconfig` group of the Oracle Identity Federation configuration. Set the `authncontext` string property in `serverconfig`, default missing.

- Set the relative path property referencing the path in the web context where the custom JSP Page or servlet resides. Set the `authnpath` string property in `serverconfig`, default `/user/loginsso`.

Use the WLST commands to set these properties in the Oracle Identity Federation configuration.

For example, the following commands, in the WLST script environment for the Oracle Identity Federation instance, configure a post-processing plug-in to be invoked after all the authentication engines:

```
setConfigProperty('serverconfig', 'authncontext', '/rootcontext', 'string')

setConfigProperty('serverconfig', 'authnpath', '/relativepath', 'string')
```

12.3.3 Example of a Post-processing Custom Action

This section shows a simple post-processing plug-in that is invoked by all the built-in authentication engines before the user is redirected to Oracle Identity Federation at the end of a local authentication operation. This plug-in accesses a custom cookie presented by the browser, extracts data from it, and sets it as Oracle Identity Federation session attributes that can then be used during the operation that creates the assertion.

Oracle Identity Federation supports the concept of session attributes set by the authentication engine during a local authentication operation:

- Oracle Identity Federation acts as an IdP
- The authentication engine flow sets some attributes as session attributes called `attr1` and `attr2`
- Oracle Identity Federation is configured to send the session attributes referenced as `attr1` and `attr2` when creating an assertion for specific service provider partners

This sample shows how a post-processing plug-in can set session attributes in a local authentication flow where a built-in authentication engine is used.

In this sample, the plug-in adds the following attributes, extracted from a custom cookie that is previously set by another component, after a successful authentication:

- `cookie-language`, containing the preferred language of the user
- `cookie-homepage`, containing the preferred home page of the user

12.3.3.1 Set-up

A custom component sets the cookie used in this example.

12.3.3.2 Packaging

The post-processing plug-in consists of a Web application with a root context set to `/plugin`, and contains one JSP page, `cookieextract.jsp`, which extracts the data from the custom cookie and set it as session attributes; the plug-in redirects the user to the federation server by means of an internal forward to resume the flow.

12.3.3.3 Oracle Identity Federation Configuration

To configure Oracle Identity Federation to invoke the post-processing plug-in before the Oracle Identity Federation server at the end of local authentication flow, take these steps:

1. Enter the WLST script environment for the Oracle Identity Federation instance.
2. Set the `authncontext` property containing the root context of the post-processing plug-in page:

   ```
   setConfigProperty('serverconfig', 'authncontext', '/plugin', 'string')
   ```

3. Set the `authnpath` property containing the relative path of the post-processing plug-in page:

   ```
   setConfigProperty('serverconfig', 'authnpath', '/cookieextract.jsp', 'string')
   ```

4. Exit the WLST script environment.

12.3.3.4 Implementation of cookieextract.jsp

The JSP looks like this:

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.util.*, javax.naming.*,
javax.naming.directory.*, java.net.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

// check if authentication was successful
if (request.getAttribute("oracle.security.fed.authn.authntime") != null)
{
// authentication was successful. Attributes will be added
Map attributes =
(Map)request.getAttribute("oracle.security.fed.authn.attributes");
if (attributes == null)
{
attributes = new HashMap();
request.setAttribute("oracle.security.fed.authn.attributes", attributes);
}

// get the cookie
Cookie[] cookies = request.getCookies();
String cookieValue = null;
for(int i = 0; i < cookies.length; i++)
{
Cookie cookie = cookies[i];
if (cookie.getName().equals("customcookie"))
cookieValue = cookie.getValue();
}
if (cookieValue != null && cookieValue.length() > 0)
{
StringTokenizer st = new StringTokenizer(cookieValue, "+");
String language = st.nextToken();
String homepage = st.nextToken();

Set languageValues = new HashSet();
languageValues.add(language);
attributes.put("cookie-language", languageValues);
```

```
Set homepageValues = new HashSet();
homepageValues.add(homepage);
attributes.put("cookie-homepage", homepageValues);
}
}

// forward to the OIF server to resume the flow
request.getSession().getServletContext().getContext("/fed").getRequestDispatcher("
/user/loginsso").forward(request, response);
%>
```

12.4 Pre-processing Custom Action for SP Integration Engine

The pre-processing plug-in is a module to which the user is directed, as part of an SP SSO flow from an SP integration module, instead of being redirected to the Oracle Identity Federation server to start the Federation SSO operation. The plug-in enables custom actions to be taken before the SSO flow.

When the pre-processing plug-in is in use, the SP integration module internally forwards the user to the plug-in, supplying the runtime data to be used for the federated SSO operation. The plug-in performs custom operations, and forwards the user to Oracle Identity Federation - with the runtime data - to resume the Federation SP SSO flow.

12.4.1 Implementing the Pre-processing Plug-in

When the SP engine redirects a user to Oracle Identity Federation, it passes certain data to the server as attributes on the `HttpServletRequest` object. The same data is made available to the pre-processing plug-in.

Here is the data passed to the plug-in:

- An optional Provider ID referencing the IdP to use for the federated SSO. If missing, Oracle Identity Federation uses the IdP mapped for the specified authentication mechanism. If no IdP could be found, Oracle Identity Federation uses the IdP configured as the default SSO IdP (String, identified by oracle.security.fed.sp.providerid)
- An optional federation ID referencing the affiliation to use to trigger the federated SSO (String, identified by oracle.security.fed.sp.federationid)
- The relay state containing a small string, for example a reference to some data saved in a repository or a small URL pointing to the protected resource to which the user is redirected after completion of the SSO operation. (String, identified by oracle.security.fed.sp.relaystate)
- The protected resource the user was trying to access, when the SP Engine is an out-of-the-box SP engine (either Oracle Access Manager, Oracle Single Sign-On, or test SP engine). (identified by `oracle.security.fed.sp.returnurl`)
- The identifier of the SP engine that started the SSO flow (String, identified by oracle.security.fed.sp.engineid)
- An optional flag indicating if Oracle Identity Federation should authenticate the user locally using the authentication engines or if a federated SSO should be started by redirecting the user to an IdP for authentication (Boolean, identified by oracle.security.fed.sp.localauthn; default is `false`)

- Whether to use the configuration stored in Oracle Identity Federation or to only start the SSO based on the information being passed by the SP engine, except the IdP (Boolean, identified by oracle.security.fed.sp.usedefault). If missing, `true` is assumed.
- Whether the SP should ask the IdP to challenge the user even if he/she is already authenticated (Boolean, identified by oracle.security.fed.sp.forceauthn). This parameter is ignored if `usedefault` is `true` or missing.
- Whether the SP should allow the IdP to create a federation record, if one does not yet exist, during the SSO operation (Boolean, identified by oracle.security.fed.sp.allowfedcreation). This parameter is ignored if `usedefault` is `true` or missing.
- Whether the SP should ask the IdP not to interact with the user during the SSO operation (Boolean, identified by oracle.security.fed.sp.passive). This parameter is ignored if `usedefault` is `true` or missing.
- The binding to use when sending the `AuthnRequest` (String, identified by oracle.security.fed.sp.requestbinding). This parameter is ignored if `usedefault` is `true` or missing. Acceptable values are `httpredirect`, `httppost`, and `httppostsimple` depending on the protocol.
- The binding to use when sending the response with the assertion (String, identified by oracle.security.fed.sp.responsebinding). This parameter is ignored if `usedefault` is `true` or missing. Acceptable values are artifact or httpost depending on the protocol.
- An optional authentication mechanism comparison specifying to the SP which authentication context comparison to request the IdP to use during authentication. (String, identified by oracle.security.fed.sp.authnmechcomparison). This parameter is ignored if `usedefault` is `true` or missing.
- The NameID format the SP will issue to the IdP for the SSO operation (String, identified by oracle.security.fed.sp.nameidformat). This parameter is ignored if `usedefault` is `true` or missing.
- Optional attributes to be requested from the IdP during the Federation SSO operation, for example when interacting with an OpenID IdP. The data is passed as a `Map` with `String`s as keys and set of `Object`s as values (identified by `oracle.security.fed.sp.attributes`). The values are optional, while the keys contain the attribute names.

  See Also: Section 10.4.2, "Developing and Implementing the Integration Module" for details about the data provided by the SP integration engine.

  Notes:

  - The plug-in must provide Oracle Identity Federation server with the data that was passed to it as part of the SP federated SSO flow; this consists of attributes that were set on the `HttpServletRequest` object.
  - The plug-in can modify all attributes that were set on the `HttpServletRequest` object except the oracle.security.fed.sp.engineid attribute.

12.4.2 Configuring Oracle Identity Federation for the Plug-in

To configure Oracle Identity Federation to forward the user to a pre-processing plug-in at the start of the SP federated SSO flow, set the following properties:

- Create or set the web context property, referencing the web context where the custom JSP Page or servlet resides. This property is located in the `serverconfig` group of the Oracle Identity Federation configuration. Set the `spcontext` string property in `serverconfig`, default `missing`.
- Set the relative path property referencing the path in the web context where the custom JSP Page or servlet resides. Set the `sppath` string property in `serverconfig`, default `/sp/startsso`.

Use the WLST commands to set these properties in the Oracle Identity Federation configuration.

For example, the following commands, in the WLST script environment for the Oracle Identity Federation instance, configure a pre-processing plug-in to be invoked prior to the SSO flow:

```
setConfigProperty('serverconfig', 'spcontext', '/rootcontext', 'string')

setConfigProperty('serverconfig', 'sppath', '/relativepath', 'string')
```

12.4.3 Example of a Pre-processing Plug-in

This example shows a simple pre-processing plug-in, invoked by the out-of-the-box SP engines before the user is forwarded to Oracle Identity Federation to start the Federation SSO operation, to determine the IdP to be used for the Federation SSO operation.

In this example, the local domain has two resources protected by Oracle Access Manager:

- http://www.domain.com/resource1, and the IdP to use must be idp1.com
- http://www.domain.com/resource2, and the IdP to use must be idp2.com

12.4.3.1 Setup

The Oracle Identity Federation/SP server is integrated with Oracle Access Manager.

12.4.3.2 Packaging

The pre-processing plug-in consists of a Web application with a root context set to `/plugin`, and contains one JSP page, `fedidpeval.jsp`, which evaluates the URL of the protected resource and determines which IdP to use.

12.4.3.3 Configuring Oracle Identity Federation

To configure the Oracle Access Manager SP engine to invoke the pre-processing plug-in before Oracle Identity Federation processing at the beginning of the federation SSO flow, take these steps:

1. Enter the WLST script environment for the Oracle Identity Federation instance.
2. Set the `spcontext` property containing the root context of the pre-processing plug-in page:

   ```
   setConfigProperty('serverconfig', 'spcontext', '/plugin', 'string')
   ```

3. Set the `sppath` property containing the relative path of the pre-processing plug-in page:

   ```
   setConfigProperty('serverconfig', 'sppath', '/fedusercheck.jsp', 'string')
   ```

4. Exit the WLST script environment.

Implementation of fedusercheck.jsp

```
Implementation of fedusercheck.jsp
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.util.*, javax.naming.*,
javax.naming.directory.*, java.net.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

String returnURL = (String)
request.getAttribute("oracle.security.fed.sp.returnurl");
String providerid = null;

if (returnURL != null && returnURL.startsWith("http://www.domain.com/resource1"))
{
providerid = "idp1.com";
}
else if (returnURL != null &&
returnURL.startsWith("http://www.domain.com/resource2"))

{
providerid = "idp2.com";
}

if (providerid != null)
request.setAttribute("oracle.security.fed.sp.providerid", providerid);

// forward to OIF
request.getSession().getServletContext().getContext("/fed").getRequestDispatcher("
/sp/startsso").forward(request, response);
%>
```

12.5 Post-processing Custom Action for SP Engine

The post-processing plug-in is a module to which the user is directed, as part of a federated SSO operation. The plug-in enables custom actions to be taken after the operation is complete.

When the plug-in is in use, Oracle Identity Federation does not redirect the user to the SP engine; rather, it forwards the user internally to the plug-in, passing it the data resulting from the SSO operation. After performing its custom actions, the plug-in forwards the user to the correct SP engine, along with the data originally provided by the federation server, to resume the authentication flow.

12.5.1 Implementing the Post-processing Plug-in

When Oracle Identity Federation redirects the user to the SP engine at the end of the federated SSO flow, it passes certain data to the engine as attributes on the

`HttpServletRequest` object. The same data is made available to the post-processing plug-in.

Here is the data passed to the plug-in:

- Whether the SSO operation was successful (Boolean, identified by oracle.security.fed.sp.authnresult)
- The identifier of the user (String, identified by oracle.security.fed.sp.userid)
- Authentication time (Date, identified by oracle.security.fed.sp.authntime)
- Expiration time of the authenticated session (Date, identified by oracle.security.fed.sp.expirationtime)
- The authentication mechanism used to identify the user (String, identified by oracle.security.fed.sp.authnmech)
- The relay state (String, identified by oracle.security.fed.sp.relaystate)
- The protected resource the user was trying to access, when the SP engine is an out-of-the-box SP engine (Oracle Access Manager, Oracle Single Sign-On, or test SP engine); identified by oracle.security.fed.sp.returnurl.
- The contents of the assertion consisting of the NameID (the assertion issuer) and the optional attributes.

  The assertion content is not passed as XML data, that is, the original assertion is not passed back to the module; rather, the data is passed as a map with strings as keys and a set of objects as values (identified by oracle.security.fed.sp.attributes). The extra data is referenced as:

  – orafed-nameid-value – the Name ID value
  – orafed-nameid-qualifier – the Name ID qualifier
  – orafed-nameid-format – the Name ID format
  – orafed-providerid – the ProviderID
  – orafed-assertionid - the ID of the assertion
  – orafed-xmlmessage - the optional XML message containing the assertion

  See Also : Section 6.13.2, "Providing XML Message to SP Engine after SSO Completes" for details.

- The top status of the SAML response (String, identified by oracle.security.fed.sp.topstatus)
- The low status of the SAML response if any, (String, identified by oracle.security.fed.sp.lowstatus)
- The status message if any (String, identified by oracle.security.fed.sp.statusmessage)
- The ProviderID that created the SSO assertion (String, identified by oracle.security.fed.sp.providerid)
- The identifier of the SP engine to process this information (identified by oracle.security.fed.sp.engineid)
- The Oracle Identity Federation identifier of the user session. (String, identified by oracle.security.fed.sessionid)

Oracle Identity Federation passes the sessionID of the user session to the SP engine so it can persist the state linked to the user, and can reference the data using the sessionID value. Later, when the logout flow is executed, Oracle Identity Federation passes the `sessionID` being logged out to the engine, so that the engine can delete the data used for this user session.

Notes:

- The plug-in must provide the SP engine with the data that was passed to it as part of the SP federated SSO flow; this consists of attributes that were set on the `HttpServletRequest` object.
- The plug-in can modify all attributes that were set on the `HttpServletRequest` object except the oracle.security.fed.sp.engineid attribute.

12.5.2 Configuring Oracle Identity Federation for the Plug-in

Configure Oracle Identity Federation to forward the user to a post-processing plug-in rather than to the SP engine at the end of the SSO flow by performing these tasks:

- identify the SP engine whose flow will be modified. Choices are: Oracle SSO, Oracle Access Manager, Test SP, Proxy engines.
- Set the following two properties:
  - Create or set the web context property, referencing the web context where the custom JSP page or servlet of the post-processing plug-in resides. This property is located in the `spengines` group of the Oracle Identity Federation configuration.
  - the relative path property referencing the path in the web context where the custom JSP page or servlet resides.

Table 12–2 shows how to set the properties for each SP engine:

Table 12–3 SP Engine Configuration for Post-processing Plug-in

SP Engine	web context property	relative path property
Oracle Single Sign-On	osso-login-context (default missing)	osso-login (default /user/spretosso)
Oracle Access Manager 10*g*	oam-login-context (default missing)	oam-login (default /user/spretoam)
Oracle Access Manager 11*g*	oam11g-login-context	oam11g-login
TestSP	testsp-login-context (default missing)	testsp-login (default /user/testspretsso)
Fed SSO Proxy	proxy-login-context (default missing)	proxy-login (default /user/proxyretsso)

Use the WLST commands to set these properties in the Oracle Identity Federation configuration.

For example, the following commands, in the WLST script environment for the Oracle Identity Federation instance, configure a post-processing plug-in to be invoked before the Oracle SSO SP engine at the end of the federated SSO operation:

```
setConfigProperty('spengines', 'osso-login-context', '/rootcontext', 'string')

setConfigProperty('spengines', 'osso-login', '/relativepath', 'string')
```

12.5.3 Example of a Post-processing Plug-in

This section shows a simple post-processing plug-in to be invoked by Oracle Identity Federation, at the end of a federated SSO and before the Oracle Access Manager SP Engine, to check that the user referenced in the assertion actually belongs to the local domain of the identity provider (IdP).

Oracle Identity Federation can have multiple IdPs as federation partners, each authenticating users on behalf of the Oracle Identity Federation SP and the local domain where the federation server is deployed.

In a typical scenario:

- the user is redirected to the IdP for authentication
- the IdP creates an assertion containing the user's identity
- Oracle Identity Federation/SP maps the incoming assertion to a local user record and creates an authenticated session in the domain.

Some deployments might require Oracle Identity Federation/SP to verify that the user referenced in the assertion belongs to the domain of the assertion issuer; that way, users may only be able to access the services of the SP domain if they used their primary identity providers.

In this example, the assertion Name ID is based on the email address, with the address domain being mapped to a single IdP. The following IdPs are known to Oracle Identity Federation/SP:

- `http://idp.acme.com/fed/idp`, and the corresponding email addresses of the users from this domain is `@acme.com`
- `http://samlidp.foo.com/fed/idp`, and the corresponding email addresses of the users from this domain is `@foo.com`
- `http://fed.bar.com/fed/idp`, and the corresponding email addresses of the users from this domain is `@bar.com`

This section contains these topics:

12.5.3.1 Set-up

The Oracle Identity Federation/SP server is integrated with Oracle Access Manager.

12.5.3.2 Packaging

The post-processing plug-in consists of a Web application with a root context set to `/plugin`, and contains one JSP page, `fedusercheck.jsp`, which verifies the assertion data received from the IdP. If the verification is successful the plug-in redirects the user to the Oracle Access Manager SP engine through an internal forward, so that an Oracle Access Manager session can be created, and the user redirected to the protected resource.

12.5.3.3 Oracle Identity Federation Configuration

To configure Oracle Identity Federation to invoke the post-processing plug-in before the Oracle Access Manager SP engine at the end of the federated SSO flow, take these steps:

1. Enter the WLST script environment for the Oracle Identity Federation instance.
2. Set the `oam-login-context` property containing the root context of the post-processing plug-in page:

   ```
   setConfigProperty('spengines', 'oam11g-login-context', '/plugin', 'string')
   ```

3. Set the `oam-login` property containing the relative path of the post-processing plug-in page:

   ```
   setConfigProperty('spengines', 'oam11g-login', '/fedusercheck.jsp', 'string')
   ```

4. Exit the WLST script environment.

12.5.3.4 Implementation of fedusercheck.jsp

The JSP is implemented as follows:

```
<%@page buffer="5" autoFlush="true" session="false"%>
<%@page language="java" import="java.util.*, javax.naming.*,
javax.naming.directory.*, java.net.*"%>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1969 17:04:19 GMT");

Map attributes = (Map)request.getAttribute("oracle.security.fed.sp.attributes");
String email = (String)attributes.get("orafed-nameid-value");
String providerid = (String)attributes.get("orafed-providerid");

// check that email address is linked to providerid
if ("http://idp.acme.com/fed/idp".equals(providerid))
{
// acme domain
        if (!email.endsWith("@acme.com"))
        {
                throw new Exception("User is invalid: " + email + " does not
belong to " + providerid);
        }
}
else if ("http://samlidp.foo.com/fed/idp".equals(providerid))
{
// foo domain
        if (!email.endsWith("@foo.com"))
        {
                throw new Exception("User is invalid: " + email + " does not
belong to " + providerid);
        }
}
else if ("http://fed.bar.com/fed/idp".equals(providerid))
{
// bar domain
        if (!email.endsWith("@bar.com"))
        {
                throw new Exception("User is invalid: " + email + " does not
belong to " + providerid);
```

```
        }
}
else
        throw new Exception("Unknown IdP: " + providerid);

// forward to the OAM SP Engine to resume the flow
request.getSession().getServletContext().getContext("/fed").getRequestDispatcher("
/user/spretoam11g").forward(request, response);
%>
```

Part IV

Appendices

This part contains the following reference appendices:

A Oracle Identity Federation MBeans

Several MBeans manage the underlying configuration of the Oracle Identity Federation server. The configuration data is stored in three files:

- `config.xml` contains server-wide configuration.
- `cot.xml` stores provider-specific configuration.
- `datastore.xml` stores back-end data store configuration.

This appendix describes the function of each MBean and the corresponding configuration file elements, and contains these sections:

A.1 Server-wide Configuration (config.xml)

`FederationConfig`, `Config`, `PropertiesMap`, and `PropertiesList` MBeans manage server wide configuration in `config.xml`.

- FederationConfig
- Config
- PropertiesList
- PropertiesMap

A.1.1 FederationConfig

This section describes the `FederationConfigMXBean` and its corresponding `FederationConfig` element.

- FederationConfigMXBean
- The FederationConfig Element

A.1.1.1 FederationConfigMXBean

The `FederationConfigMXBean` manages the sequence of `Config` elements and the life cycle of their corresponding `ConfigMXBeans`. It exposes the following operations:

- createEmptyConfig: Given a name, creates a new `Config` element and a corresponding `ConfigMXBean`. The given name cannot be null or the empty string, and it must be unique across all `Config` elements in this `FederationConfig`
- destroyConfig: Given a name, destroys the `Config` element with the given name and un-registers its corresponding `ConfigMXBean`.
- hasConfig: Given a name, returns `true` if and only if there exists a `Config` element in this Federation `Config` with the given name.
- retrieveConfig: Given a name, returns the `ObjectName` with which the `ConfigMXBean` corresponding to the given `Config` element is registered in the MBean server.
- retrieveConfigs: Returns the `ObjectNames` with which the `ConfigMXBeans` corresponding to all child `Config` elements are registered in the MBean server.

A.1.1.2 The FederationConfig Element

`FederationConfig` is the top element of the `config.xml` file. It contains a sequence of `Config` elements.

```
<fed:FederationConfig xmlns:fed="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd">
        <fed:Config name="serverconfig">
…
        </fed:Config>
        <fed:Config name="idpglobal">
…
        </fed:Config>
 …
</fed:FederationConfig>
```

A.1.2 Config

This section describes the `ConfigMXBean` and its corresponding `Config` element.

- ConfigMXBean
- The Config Element

A.1.2.1 ConfigMXBean

The `ConfigMXBean` manages the addition, removal and retrieval of properties, and manages its `PropertiesList` and `PropertiesMap` children by managing the life cycle of their corresponding `PropertiesListMXBeans` and `PropertiesMapMXBeans`. It exposes the following operations:

- element name retrieval
- retrieval, addition, and removal of properties
- life cycle management of `PropertiesListMXBeans`
- life cycle management of `PropertiesMapMXBeans`

Element Name Retrieval

`getName` retrieves the name of its corresponding `Config` element.

Retrieval, Addition, and Removal of Properties

Operations to manage addition, removal, and retrieval of properties are:

- hasProperty: Given a name, returns `true` if and only if there exists a `Property` in this `Config` with the given name.
- putProperty: Given a name, a value, and a type, adds a `Property` to this `Config` with the specified name, value and type.

  If there already exists a `Property` with the specified name, sets the value and type of the existing property to the given ones. However, if there already exists a `Property` with the given name, the given type must match the type of the existing property.

  In either case, the type must be one of: '`string`', '`boolean`', '`long`', and the value must be of the specified type. The name cannot be `null` or the empty string.
- removeAllProperties: Removes all `Property` elements in this `Config`.
- removeProperty: Given a name, removes the `Property` with the given name from this `Config`.
- retrievePropertyType: Given a name, returns the type of the `Property` in this `Config` with the given name.
- retrievePropertyValue: Given a name, returns the type of the `Property` in this `Config` with the given name.

Manage Life Cycle of PropertiesListMXBeans

Operations for life cycle management of `PropertiesListMXBeans` include:

- createPropertiesList: Given a name, creates a new `PropertiesList` element and a corresponding `PropertiesListMXBean`. The given name cannot be `null` or the empty string and it must be unique across all `PropertiesList` elements in this `Config`.
- destroyAllPropertiesLists: Destroys all `PropertiesList` elements and unregisters their corresponding `PropertiesListMXBeans`.
- destroyPropertiesList: Given a name, destroys the `PropertiesList` element in this `Config` with the given name, and unregisters its corresponding `PropertiesListMXBean`.
- hasPropertiesList: Given a name, returns `true` if and only if there exists a `PropertiesList` in this `Config` with the given name.
- retrieveAllPropertiesLists: Returns the `ObjectNames` with which the `PropertiesListMXBeans` corresponding to all child `PropertiesList` elements are registered in the MBean server.
- retrievePropertiesList: Given a name, retrieves the `ObjectName` with which the `PropertiesListMXBean` corresponding to the `PropertiesList` element in this `Config` with the given name is registered in the MBean server.

Manage Life Cycle of PropertiesMapMXBeans

Operations for life cycle management of `PropertiesMapMXBeans` are equivalent to those that manage the life cycle of child `PropertiesListMXBeans`.

A.1.2.2 The Config Element

Config elements have a name attribute and `Property`, `PropertiesList`, and `PropertiesMap` elements as children:

```
<fed:FederationConfig xmlns:fed="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd">
       <fed:Config name="serverconfig">
```

```
…
        </fed:Config>
        <fed:Config name="idpglobal">
<fed:Property name="providerid" type="string">
http://node1.us.example.com:1234/fed/idp</fed:Property>
<fed:Property name="lib11enabled" type="boolean">true</fed:Property>
<fed:PropertiesList name="sendattributefornameid">
 …
</fed:PropertiesList>
<fed:PropertiesMap name="attr-value-filters">
…
</fed:PropertiesMap>
                <fed:PropertiesMap name="attr-value-mappings">
…
</fed:PropertiesMap>
 </fed:Config>
 …
</fed:FederationConfig>
```

A.1.3 PropertiesList

This section describes the `PropertiesListMXBean` and its corresponding `PropertiesList` element.

- PropertiesListMXBean
- The PropertiesList Element

A.1.3.1 PropertiesListMXBean

A `PropertiesListMXBean` manages the addition, removal, and retrieval of properties at a given index. It exposes the following operations:

- addProperty (overloaded): Given a value, a type and an index, adds a `Property` with the specified name and type at the given index. The type must be one of: 'string', 'boolean', 'long', and the value must be of the type specified.
- addProperty (overloaded): Given a value and a type, adds a `Property` with the specified name and type to the end of this `PropertiesList`. The type must be one of: 'string', 'boolean', 'long', and the value must be of the type specified.
- getName: Returns the name of this `PropertiesList`.
- hasPropertyValue: Given a value, returns `true` if and only if there exists a `Property` in this `PropertiesList` with the given value.
- indexOf: Given a value, returns the index of the first `Property` that has the specified value, or -1 if no `Property` in this `PropertiesList` has the specified value.
- removeAllProperties: Removes all `Property` elements from this `PropertiesList`.
- removeProperty (overloaded): Given an index, removes the `Property` element at the given index.
- removeProperty (overloaded): Given a value, removes the first `Property` element that has the specified value.
- retrieveAllPropertyValues: Returns a list containing the values of the Property elements in this PropertiesList. The values are returned in the same order in which the Property elements appear.

- retrieveNumberOfProperties: Returns the number of `Property` elements in this PropertiesList.
- retrievePropertyType: Given an index, returns the type of the `Property` element at the given index.
- retrievePropertyValue: Given an index, returns the value of the `Property` element at the given index.

A.1.3.2 The PropertiesList Element

A `PropertiesList` has a name attribute and `Property` elements as children. `Property` elements inside a `PropertiesList` do not have names.

```
<fed:FederationConfig xmlns:fed="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd">
        <fed:Config name="serverconfig">
…
        </fed:Config>
        <fed:Config name="idpglobal">
 …
 </fed:Config>
 <fed:Config name="idpliberty11">
        <fed:PropertiesList name="ssobindings">
                <fed:Property type="string">artifact</fed:Property>
                <fed:Property type="string">httppost</fed:Property>
        </fed:PropertiesList>
        <fed:PropertiesList name="authnreqbindings">
                <fed:Property type="string">httppost</fed:Property>
                <fed:Property type="string">httpredirect</fed:Property>
        </fed:PropertiesList>
 …
 </fed:Config>
 …
</fed:FederationConfig>
```

A.1.4 PropertiesMap

This section describes the `PropertiesMapMXBean` and its corresponding `PropertiesMapMXBean` element.

- PropertiesMapMXBean
- The PropertiesMap Element

A.1.4.1 PropertiesMapMXBean

A `PropertiesMapMXBean` manages the addition, removal and retrieval of properties, and manages its `PropertiesList` and `PropertiesMap` children by managing the life cycle of their corresponding `PropertiesListMXBeans` and `PropertiesMapMXBeans`. It exposes the same operations as a `ConfigMXBean`, with the addition of the following operation:

retrieveAllPropertyNames: Returns a list containing the names of the `Property` elements in this `PropertiesMap`.

A.1.4.2 The PropertiesMap Element

`PropertiesMap` elements have a name attribute and `Property`, `PropertiesList`, and `PropertiesMap` elements as children.

```
<fed:FederationConfig xmlns:fed="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd">
```

```
…
<fed:Config name="fedusersearch">
<fed:PropertiesMap name="fedldap">
<fed:Property name="includesearchattrs" type="boolean">true</fed:Property>
<fed:Property name="defaultsorton"
type="string">orclFedOwnerGUID</fed:Property>
               <fed:PropertiesList name="defaultsearch">
                …
               </fed:PropertiesList>
               <fed:PropertiesList name="defaultdisplay">
                …
               </fed:PropertiesList>
<fed:PropertiesMap name="displaynames">
                …
               </fed:PropertiesMap>
        </fed:PropertiesMap>
 …
</fed:Config>
…
</fed:FederationConfig>
```

A.2 Provider-specific Configuration

`CircleOfTrust`, `PeerProvider` MBeans support provider-specific configuration in `cot.xml`.

- CircleOfTrust
- PeerProvider

A.2.1 CircleOfTrust

This section describes the `CircleOfTrustMXBean` and its corresponding `CircleOfTrust` element.

- CircleOfTrustMXBean
- The CircleOfTrust Element

A.2.1.1 CircleOfTrustMXBean

The `CircleOfTrustMXBean` manages the sequence of `PeerProvider` elements and the life cycle of their corresponding `PeerProviderMXBeans`. It exposes the following operations:

- createPeerProvider: Given a description, provider ID, provider type, and version, creates a new `PeerProvider` element and a corresponding `PeerProviderMXBean`. None of the parameters passed can be `null`, and the provider ID, provider type, and version cannot be the empty string. If there already exists a `PeerProvider` with the given provider ID, the existing provider is destroyed and replaced by the new provider.
- destroyPeerProvider: Given a provider ID, destroys the `PeerProvider` element in this `CircleOfTrust` with the given provider ID, and unregisters its corresponding `PeerProviderMXBean` from the MBean server.
- hasPeerProvider: Given a provider ID, returns `true` if and only if there exists a `PeerProvider` element in this `CircleOfTrust` with the given provider ID.

- loadMetadata: Given a `String` with a Peer Provider's metadata, creates a new `PeerProvider` element with the information found in the metadata and creates a corresponding `PeerProviderMXBean`. The metadata cannot be `null` and it must be in XML format. The metadata must also comply to SAML 1.x, SAML 2.0, or Liberty 1.x specifications.
- retrievePeerProvider: Given a provider ID, returns the `ObjectName` with which the `PeerProviderMXBean` corresponding to the `PeerProvider` element in this `CircleOfTrust` with the given provider ID, is registered on the MBean server.
- retrievePeerProviders: Returns the `ObjectNames` with which the `PeerProviderMXBeans` corresponding to all child `PeerProvider` elements are registered in the MBean server.

A.2.1.2 The CircleOfTrust Element

`CircleOfTrust` is the top element of the `cot.xml` file. It contains a sequence of `PeerProvider` elements:

```
<CircleOfTrust xmlns="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd">
…
 <PeerProvider version="SAML1.0" succinctID="Iyrw+aKYfAkLFKROZCE2qe2w0Qk="
providerType="idp" providerID="http://node2.us.example.com:1234/fed/idp"
description="">
        …
</PeerProvider>
    <PeerProvider version="SAML2.0" succinctID="AZh2wC8biWp6uPwO4KgKLY82EQ8="
providerType="idp" providerID="http://node3.us.example.com:1234/fed/idp"
description="">
…
</PeerProvider>
…
</CircleOfTrust>
```

A.2.2 PeerProvider

This section describes the `PeerProviderMXBean` and its corresponding `PeerProvider` element.

- PeerProviderMXBean
- The PeerProvider Element

A.2.2.1 PeerProviderMXBean

The `PeerProviderMXBean` manages the retrieval and setting of attributes and text content of the `Metadata` element. It also manages the life cycle of the `Config` element's corresponding `ConfigMXBean`. It exposes the following operations:

- retrieving and setting attributes
- retrieving and setting child `Metadata` element
- life cycle management of child `ConfigMXBean`

Retrieving and setting of attributes

Operations to retrieve and set attributes include:

- get/setDescription: gets/sets the value of the description attribute. The value to be set cannot be `null`.

- getProviderID: gets the value of the provider ID attribute.
- get/setProviderType: gets/sets the value of the provider type attribute. The value to be set cannot be `null` or the empty string.
- get/setVersion: gets/sets the value of the version attribute. The value to be set cannot be `null` or the empty string.
- get/setSuccinctID: gets/sets the value of the succinct ID attribute. The value to be set cannot be `null` or the empty string.

Retrieving and setting of child Metadata element

Operations to retrieve and set the `Metadata` element include:

- retrieveMetadata: Returns a `String` containing this Peer Provider's metadata in XML format.
- updateMetadata: Given a `String` containing metadata, sets the text value of this Peer Provider's `Metadata` element to the given metadata. The given metadata must be in XML format.

Life cycle management of child ConfigMXBean

`retrieveConfig` returns the `ObjectName` with which the `ConfigMXBean` corresponding to the `Config` element in this `PeerProvider` is registered in the MBean server.

A.2.2.2 The PeerProvider Element

`PeerProvider` elements have the following attributes: description, provider ID, provider type, version, and succinct ID.   They also have a single `Metadata` element and a single `Config` element as child elements.

```
<CircleOfTrust xmlns="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd">
 …
<PeerProvider version="SAML2.0" succinctID="AZh2wC8biWp6uPwO4KgKLY82EQ8="
providerType="idp" providerID="http://node3.us.example.com:1234/fed/idp"
description="">
<Metadata>
        …
<Metadata>
<Config name="http://node2.us.example.com:1234/fed/idp">
…
</Config>
</PeerProvider>
…
</CircleOfTrust>
```

A.3 Data-store Configuration

`Datastore` and `DiscoveryProvider` MBeans manage configuration of data stores in `data-store.xml`.

- Datastore
- DiscoveryProvider

A.3.1 Datastore

This section describes the `DatastoreMXBean` and its corresponding datastore element.

- DatastoreMXBean
- The datastore Element

A.3.1.1 DatastoreMXBean

The `DatastoreMXBean` manages the retrieval of the `defaultPackage` attribute and also manages the sequence of `DiscoveryProvider` elements by controlling their corresponding `DiscoveryProviderMXBeans`. It exposes the following operations:

- getDefaultPackage: returns the value of the `defaultPackage` attribute.
- createDiscoveryProvider: Given a type, setter, classname, and dependsOn, creates a new `DiscoveryProvider` element and a corresponding `DiscoveryProviderMXBean`. None of the parameters can be `null`, and the classname and setter cannot be the empty string.   The type must be unique across all `DiscoveryProvider` elements in this datastore.
- destroyDiscoveryProvider: Given a type, destroys the `DiscoveryProvider` element with the given type, and unregisters its corresponding `DiscoveryProviderMXBean`.
- hasDiscoveryProvider: Given a type, returns `true` if and only if there exists a `DiscoveryProvider` in this datastore with the given type.
- retrieveDiscoveryProvider: Given a type, returns the `ObjectName` with which the `DiscoveryProviderMXBean` corresponding to the `DiscoveryProvider` in this datastore with the given type is registered in the MBean server.
- retrieveDiscoveryProviders: Returns the `ObjectNames` with which the `DiscoveryProviderMXBeans` corresponding to all child `DiscoveryProvider` elements are registered in the MBean server.

A.3.1.2 The datastore Element

`datastore` is the top element of the `data-store.xml` file. It has a `defaultPackage` attribute and it contains a sequence of `DiscoveryProvider` elements:

```
<datastore xmlns="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd"
defaultPackage="oracle.security.fed.jvt.discovery.model">
    <DiscoveryProvider type="ActiveIdentityProviderFederationDiscovery">
        …
    </DiscoveryProvider>
    <DiscoveryProvider type="ActiveServiceProviderFederationDiscovery">
        …
    </DiscoveryProvider>
     …
    <DiscoveryProvider type="ConfigurationDiscovery">
        …
    </DiscoveryProvider>
     …
</datastore>
```

A.3.2 DiscoveryProvider

This section describes the `DiscoveryProviderMXBean` and its corresponding `DiscoveryProvider` element.

- DiscoveryProviderMXBean
- The DiscoveryProvider Element

A.3.2.1 DiscoveryProviderMXBean

The `DiscoveryProviderMXBean` manages the retrieval and setting of attributes and of the text content of the `ClassName` element. It also manages the sequence of `DiscoveryProvider` elements in its child `Dependencies` element by managing the life cycle of their corresponding `DiscoveryProviderMXBeans`. It contains operations to:

- manage retrieval and setting of attributes
- manage retrieval and setting of child `ClassName` elements
- manage the life cycle of grandchild `DiscoveryProviderMXBeans`

Retrieve and Set Attributes

Operations to retrieve and set attributes include:

- getDependsOn: Returns the value of the `dependsOn` attribute
- getSetter: Returns the value of the setter attribute
- getType: Returns the value of the type attribute

Retrieve and Set the Child ClassName Element

Operations to retrieve and set the child `ClassName` elements include:

- changeClassNameTo: Given a class name, sets the text value of the `ClassName` element to the given class name. The given class name cannot be `null` or the empty string.
- retrieveClassName: Returns the text value of the `ClassName` element of this `DiscoveryProvider`

Manage the Life Cycle of the Grandchild DiscoveryProviderMXBeans

Operations to manage the life cycle of `DiscoveryProviderMXBeans` include:

- createDiscoveryProviderDependency: Given a setter, class name, and dependsOn, creates a new `DiscoveryProvider` element inside this `DiscoveryProvider's` child `Dependencies` element with the given setter, class name, and dependsOn, and the type of this `DiscoveryProvider`. Also creates a corresponding `DiscoveryProviderMXBean`.
- destroyDiscoveryProviderDependency: Given a setter, destroys the `DiscoveryProvider` element in this `DiscoveryProvider's Dependencies` with the given setter, and unregisters its corresponding MBean from the MBean server.
- hasDiscoveryProviderDependency: Given a setter, returns true if and only if there exists a `DiscoveryProvider` in this `DiscoveryProvider's Dependencies` with the given setter.
- retrieveDiscoveryProviderDependencies: Returns the `ObjectNames` with which the `DiscoveryProviderMXBeans` corresponding to all child `DiscoveryProvider` elements in this `DiscoveryProvider's Dependencies` are registered in the MBean server.
- retrieveDiscoveryProviderDependency: Given a setter, returns the `ObjectName` with which the `DiscoveryProviderMXBean` corresponding to the `DiscoveryProvider` element with the given setter in this `DiscoveryProvider's Dependencies`, is registered in the MBean server.

A.3.2.2 The DiscoveryProvider Element

`DiscoveryProvider` elements have type, setter, and dependsOn attributes. They also have a single `ClassName` element and a single `Dependencies` element as children. The `Dependencies` element contains a sequence of `DiscoveryProvider` elements:

```
<datastore xmlns="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd"
defaultPackage="oracle.security.fed.jvt.discovery.model">
    …
    <DiscoveryProvider type="ConfigurationDiscovery">
<ClassName>
oracle.security.fed.jvt.discovery.model.config.ChainingConfigDiscoveryProvider
       </ClassName>
       <Dependencies>
   <DiscoveryProvider type="ConfigurationDiscovery"
setter="setConfigurationDiscovery">
              …
           </DiscoveryProvider>
     …
       </Dependencies>
    </DiscoveryProvider>
     …
</datastore>
```

A.4 Oracle Identity Federation Schema

The Oracle Identity Federation schema is as follows:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd"
            elementFormDefault="qualified" attributeFormDefault="unqualified"
            xmlns:fed="http://xmlns.oracle.com/fed/schema/oif-11_2.xsd">
  <xsd:element name="FederationConfig" type="fed:FederationConfigType"/>
  <xsd:complexType name="FederationConfigType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0"
                   ref="fed:Config"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="Config" type="fed:ConfigType"/>
  <xsd:complexType name="ConfigType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0"
                   ref="fed:Property"/>
      <xsd:element maxOccurs="unbounded" minOccurs="0"
                   ref="fed:PropertiesList"/>
      <xsd:element ref="fed:PropertiesMap" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" use="required" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="Property" type="fed:PropertyType"/>
  <xsd:complexType name="PropertyType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="name" use="required" type="xsd:string"/>
        <xsd:attribute name="type" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:element name="PropertiesList" type="fed:PropertiesListType"/>
  <xsd:complexType name="PropertiesListType">
    <xsd:choice>
      <xsd:element ref="fed:Property" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:attribute name="name" use="required" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="PropertiesMap" type="fed:PropertiesMapType"/>
  <xsd:complexType name="PropertiesMapType">
    <xsd:choice>
      <xsd:element ref="fed:Property" maxOccurs="unbounded" minOccurs="0"/>
      <xsd:element ref="fed:PropertiesList" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element ref="fed:PropertiesMap" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:choice>
    <xsd:attribute name="name" use="required" type="xsd:string"/>
  </xsd:complexType>
<xsd:element name="CircleOfTrust" type="fed:CircleOfTrustType"/>
  <xsd:complexType name="CircleOfTrustType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" ref="fed:PeerProvider"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="PeerProvider" type="fed:PeerProviderType"/>
  <xsd:complexType name="PeerProviderType">
    <xsd:sequence>
      <xsd:element ref="fed:Metadata"/>
      <xsd:element ref="fed:Config"/>
    </xsd:sequence>
    <xsd:attribute name="providerID" type="xsd:string" use="required"/>
    <xsd:attribute name="succinctID" type="xsd:string" use="required"/>
    <xsd:attribute name="description" type="xsd:string"/>
    <xsd:attribute name="providerType" type="xsd:string" use="required"/>
    <xsd:attribute name="version" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:element name="Metadata" type="fed:MetadataType"/>
  <xsd:complexType name="MetadataType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string"/>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:element name="datastore" type="fed:datastoreType"/>
  <xsd:complexType name="datastoreType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0"
                   ref="fed:DiscoveryProvider"/>
    </xsd:sequence>
    <xsd:attribute name="defaultPackage" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:element name="DiscoveryProvider" type="fed:DiscoveryProviderType"/>
  <xsd:complexType name="DiscoveryProviderType">
    <xsd:sequence>
      <xsd:element ref="fed:ClassName"/>
      <xsd:element ref="fed:Dependencies"/>
    </xsd:sequence>
    <xsd:attribute name="type" use="required" type="xsd:string"/>
    <xsd:attribute name="setter" type="xsd:string"/>
    <xsd:attribute name="dependsOn" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="ClassName" type="fed:ClassNameType"/>
  <xsd:complexType name="ClassNameType">
```

```
    <xsd:simpleContent>
      <xsd:extension base="xsd:string"/>
    </xsd:simpleContent>
  </xsd:complexType>
<xsd:element name="Dependencies" type="fed:DependenciesType"/>
  <xsd:complexType name="DependenciesType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0"
                   ref="fed:DiscoveryProvider"/>
    </xsd:sequence>
  </xsd:complexType>
```

A.5 Programmatic Access to Oracle Identity Federation MBeans

This section explains how you can remotely access the MBean server and perform operations on the MBeans.

- Access the MBean Server
- Access Oracle Identity Federation MBeans

A.5.1 Access the MBean Server

You must define certain variables when accessing the MBean server remotely:

- HOSTNAME: the hostname of the machine where Oracle Identity Federation is deployed
- PORT: Oracle Identity Federation listening port
- USERNAME and PASSWORD: the username and password of an administrator

The following code demonstrates how to access the MBean Server remotely. To run this code, you must have the following libraries in your classpath:

- `WL_HOME/server/lib/weblogic.jar`
- `WL_HOME/server/lib/wljmxclient.jar`
- `WL_HOME/server/lib/wlclient.jar`

```
MBeanServerConnection mbs = null;
try{
JMXServiceURL url = new JMXServiceURL ("t3", HOSTNAME, Integer.parseInt(PORT),
"/jndi/weblogic.management.mbeanservers.runtime");

HashMap<String, Object> env = new HashMap<String,Object>();
env.put(javax.naming.Context.SECURITY_PRINCIPAL, USERNAME);
env.put(javax.naming.Context.SECURITY_CREDENTIALS, PASSWORD);
env.put(javax.management.remote.JMXConnectorFactory.PROTOCOL_PROVIDER_PACKAGES,
"weblogic.management.remote");
JMXConnector connector = JMXConnectorFactory.connect(url, env);
mbs = connector.getMBeanServerConnection();
} catch(Exception e){ //should log exception throw new
RuntimeException(e.toString(), e);
}
```

A.5.2 Access Oracle Identity Federation MBeans

The 'top' Oracle Identity Federation Configuration MBeans are registered with the 'global/translated' versions of the `ObjectNames` shown in Table A–1 (`ObjectNames` are translated to avoid name collisions):

Table A–1 ObjectNames for Oracle Identity Federation Configuration MBeans

Configuration MBean	ObjectName
FederationConfig	com.oracle.security.fed:type=OIFConfigMBean,name=ServerConfig
CircleOfTrust	com.oracle.security.fed:type=OIFConfigMBean,name=CircleOfTrust
Datastore	com.oracle.security.fed:type=OIFConfigMBean,name=Datastore

You use queries to find the global/translated `ObjectName` of an MBean. Here is an example of a query to find the `ObjectName` of the `FederationConfigMXBean`:

```
String fedObjNameQueryString =
"com.oracle.security.fed:name=ServerConfig,type=OIFConfigMBean,*";
Set s = mbs.queryNames(new ObjectName(fedObjNameQueryString), null);
ObjectName FED_CONFIG_OBJ_NAME = null;
if(s != null && !s.isEmpty())
    FED_CONFIG_OBJ_NAME = (ObjectName)s.iterator().next();
else{
    //should log exception
    throw new RuntimeException("Cannot find FedConfigMBean");
}
```

You can operate on these MBeans directly by using the `MBeanServerConnection` `invoke` method.

> **See Also:** Java documentation on the method at
> `http://download.oracle.com/javase/1.5.0/docs/api/index.html?javax/management/MBeanServerConnection.html`

Here is an example invoking the '`retrieveConfig`' operation in the `FederationConfigMXBean`:

```
try
{
ObjectName configObjName =  (ObjectName)mbs.invoke(FED_CONFIG_OBJ_NAME,
"retrieveConfig", new Object[]{configName}, new String[]{String.class.getName()});
} catch(Exception e){
    //should log exception
    throw new RuntimeException(e.toString(), e);
}
```

After obtaining the `ObjectName` of the `ConfigMXBean`, you can perform operations in a similar manner. For example, to add a new property:

```
try
{
String previousValue = (String)mbs.invoke(configObjName, "putProperty", new
Object[]{propertyName, propertyValue, propertyType}, new
String[]{String.class.getName(), String.class.getName(), String.class.getName()});
} catch(Exception e){
    //should log exception
    throw new RuntimeException(e.toString(), e);
}
```

A.6 Oracle Identity Federation MBeans API

The Oracle Identity Federation MBeans API (javadoc) is available at:

Oracle Fusion Middleware Configuration MBean Java API Reference for Oracle Identity Federation

B Using Oracle HTTP Server as a Proxy for Oracle Identity Federation

This appendix explains how to set up Oracle HTTP Server as a proxy server for Oracle Identity Federation.

- Configuring Oracle HTTP Server as Proxy
- SSL Configuration for Oracle HTTP Server

B.1 Configuring Oracle HTTP Server as Proxy

Note: Refer to your application server documentation for more information about setting up a proxy server for your environment.

Take these steps to set up an Oracle HTTP Server as a proxy for Oracle Identity Federation:

1. If not previously created with the IdM installer, create an Oracle HTTP Server component using the following command:

   ```
   $AS_ISNT/bin/opmnctl createcomponent -componentType OHS -componentName
   $OHS_NAME
   ```

   where `$AS_ISNT` is the directory where the application server instance is installed, and `$OHS_NAME` is the name of the new Oracle HTTP Server component.

2. Edit the file `$AS_ISNT/config/OHS/$OHS_NAME/moduleconf/oif.conf`. If this file is not present, create it with this content:

   ```
   # References the WebLogic server or Cluster where OIF is running
   <Location /fed>
       # Standalone install
       # WebLogicHost myweblogic.server.com
       # WebLogicPort 7499

       # Clustered install
       # WebLogicCluster w1s1.com:7499,w1s2.com:7499,w1s3.com:7499

     SetHandler weblogic-handler
   </Location>
   ```

a. If the IdM install is in stand-alone mode, uncomment and set the `WebLogicHost` and `WebLogicPort` variables to reference the WebLogic managed server where Oracle Identity Federation is running.

```
# Standalone install
WebLogicHost OIF-HOST
WebLogicPort OIF-PORT
```

b. If the IDM install is in clustered mode, uncomment and set the `WebLogicCluster` variable to reference the WebLogic managed servers where Oracle Identity Federation is running:.

```
# Clustered install
WebLogicCluster
OIF-HOST-1:OIF-PORT-1,OIF-HOST-2:OIF-PORT-2,OIF-HOST-3:OIF-PORT-3
```

3. If using SSL from the proxy to Oracle Identity Federation, edit the `$ORACLE_HOME/ohs/conf/httpd.conf` file. Add the following directive:

```
WlSSLWallet "${ORACLE_INSTANCE}/config/${COMPONENT_TYPE}/${COMPONENT_
NAME}/keystores/default"
```

4. If you have not already done so, import the certificate of the certificate authority that issued Oracle Identity Federation certificate in this wallet. See Section 8.1, "Configuring SSL for Oracle Identity Federation" for details.
5. If using SSL with the proxy, follow the instructions in Section 8.1, "Configuring SSL for Oracle Identity Federation". Omit the section about editing the `mod_wl.conf` file.
6. Restart Oracle HTTP Server to make the configuration changes effective.

```
$AS_ISNT/bin/opmnctl restartproc process-type=OHS
```

7. Determine the proxy HTTP or HTTPS ports by going to Fusion Middleware Control, locating the Oracle HTTP Server instance, and navigating to **Administration**, then **Ports Configuration**. You can test the proxy by invoking:

```
HTTP://PROXY-HOST:PROXY_PORT/fed/sp/metadata
```

8. Reconfigure Oracle Identity Federation to use the proxy host and port for its external URLs. Locate the Oracle Identity Federation instance in Fusion Middleware Control, and navigate to **Administration**, then **Server Properties**, then **Connection Settings**:
   - Host
   - Port
   - SOAP Port
   - SSL Enabled
9. If using Oracle Access Manager as the identity management system, use the Access System console to update the Fed SSO authentication schemes. In the console, navigate to **Access System Configuration**, then **Authentication Management**. Change the **Challenge Redirect** parameter for each Oracle Identity Federation Authentication scheme to use the proxy host and port.

   See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* 10*g* for details about the Web-based user interface.

10. Communicate the changes to partners using this Oracle Identity Federation server, if necessary. Partners using SAML 2.0, SAML 1.x, or Liberty 1.x will need to download new metadata. Partners using WS-Federation will need to manually update their configurations.

11. If Oracle Identity Federation is integrated with Oracle Single Sign-On, some additional steps are required. Follow the instructions in these sections:

    - Section 3.2.2.2, "Integrate Oracle Single Sign-On with OHS"
    - Section 3.2.2.3, "Configure Oracle Identity Federation to use Oracle Single Sign-On as the Authentication Engine"
    - Section 3.2.2.4, "Configure Oracle Identity Federation for Oracle Single Sign-On SP Integration"
    - Section 3.2.2.5, "Configure Oracle Single Sign-On"

B.2 SSL Configuration for Oracle HTTP Server

To configure SSL between Oracle HTTP Server and Oracle WebLogic Server, refer to:

- *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware Securing Oracle WebLogic Server*

C Troubleshooting Oracle Identity Federation

This appendix describes common problems that you may encounter when configuring and using Oracle Identity Federation, and explains how to solve them.

C.1 Problems and Solutions

This section describes common problems and solutions arranged in these topical groups:

- General Issues
- Oracle Identity Federation Configuration Issues
- Oracle Single Sign-On Login Issues
- Oracle Access Manager Configuration Issues
- Operating System Configuration Issues
- Runtime/Single Sign-On Issues
- Performance Issues

C.1.1 General Issues

This section describes general issues and workarounds. It includes the following topics:

- Attribute Sharing with the Microsoft Internet Information Server Cannot Retrieve X.509 Certificate SubjectDN
- Signed SAML 1.0 Assertions Can Cause SSO Failures
- Encrypting Network Connections
- Connecting to an LDAP Server over SSL
- thread interrupt Messages for RDBMS Message Store
- Metadata File is Unusable when Oracle Identity Federation is Configured for SSL
- ParseException Message in Diagnostic Log

C.1.1.1 Attribute Sharing with the Microsoft Internet Information Server Cannot Retrieve X.509 Certificate SubjectDN

Problem

The attribute sharing feature cannot be used with Microsoft Internet Information Servers (IIS) with Oracle Access Manager WebGate agents installed. For this feature an authentication plugin sets an HTTP header with the SubjectDN from the client's X.509 certificate, and an authorization plug-in retrieves the header to initiate a SAML attribute query. However, because of the way the IIS WebGate performs SSL client certificate authentication, the SubjectDN header cannot be retrieved by the authorization plug-in. In this case the following error is reported at the user's browser:

```
Oracle Access Manager Operation Error Access to the URL <targetURL> has
been denied for user <OblixAnonymous user DN>.
```

Also, the following error messages are written to the `OBACCESS_INSTALL/access/oblix/config/logs/authz_attribute_plugin_log.txt` file:

```
SubjectDN header ObNullString
```

and

```
SubjectDN is missing. Assume local user and return Continue
```

Solution

There is no workaround for this problem, other than to use a different web server such as the Oracle HTTP Server or the Sun One Web Server.

C.1.1.2 Signed SAML 1.0 Assertions Can Cause SSO Failures

Problem

Because SAML 1.0 does not fully specify how the XML Signature standard is to be used, implementations of the SAML 1.0 protocol sometimes have difficulty inter-operating: an SP/RP might be unable to verify the XML Digital Signature created by a SAML 1.0 IdP.

Consequently, during a Federation SSO operation between an identity provider and a service provider using SAML 1.0 protocol, errors could be raised on the service provider server during the verification of the SAML response and the SAML assertion.

Solution

The workaround is to use the SAML 1.1 protocol instead of SAML 1.0. (In fact, one of the reasons for the SAML 1.1 revision was to allow better use of XML signatures.)

Note: Signed assertions are not required, nor are they commonly used, for the SAML 1.x SSO profiles.

C.1.1.3 Encrypting Network Connections

By default, JDBC does not encrypt network connections between Oracle Identity Federationand the Oracle Database. Sites can optionally use Oracle Advanced Security to encrypt these connections.

In configuring Oracle Identity Federation to use Oracle Internet Directory or other LDAP servers to authenticate users, a site may choose whether to use SSL to connect to the LDAP server. If you do not use SSL, unencrypted passwords may be sent over network connections between Oracle Identity Federation and the LDAP server.

C.1.1.4 Connecting to an LDAP Server over SSL

When Oracle Identity Federation needs to connect to an LDAP Server using SSL - with or without Client certificate Authentication - you first need to add the necessary certificates to the Identity and Trust Keystores in the Oracle WebLogic Server Administration Console; this information is provided on the Server/Keystores configuration screen for the managed server where Oracle Identity Federation is running.

When Oracle Identity Federation needs to connect to an LDAP Server using SSL, and you encounter connection errors, make sure that the following are true:

- Oracle WebLogic Server managed server needs to be updated to have the correct keystore/certificates, and
- the administration server where Fusion Middleware Control is running must also have the correct keystore/certificates

---

Note: If using an Oracle Internet Directory server, note that in Oracle Internet Directory, SSL with no authentication cannot be disabled, and thus SSL connections to Oracle Internet Directory might be created with no server authentication.

C.1.1.5 thread interrupt Messages for RDBMS Message Store

When an RDBMS message store is in use, you may see warnings like these in the log:

```
[2009-02-05T11:43:34.322-08:00] [wls_oif1] [WARNING] [FED-12032]
[oracle.security.fed.jvt.discovery.model.profilestate.RDBMSProfileStateDiscove
ryProvider] [tid: Thread-25] [userId: <anonymous>] [ecid:
0000Hx7vp6JESO8nvgZBF119YUEk000004,1:5576] [APP: Oracle Identity
Federation#11.1.1.1.0] [arg:
java.lang.InterruptedException: sleep interrupted] InterruptedException:
thread interrupt occurred during sleep() java.lang.InterruptedException:
sleep interrupted
```

These routine messages are part of normal operation and simply indicate that the RDBMS cleaning threads are stopped.

No action is required.

C.1.1.6 Metadata File is Unusable when Oracle Identity Federation is Configured for SSL

Problem

When the Oracle Identity Federation server is configured to use the SSL port of a proxy OHS, the metadata file appears to be corrupted when attempting to open or load the metadata (using Fusion Middleware Control, on the federation server's security and trust page, metadata tab.

Solution

When Oracle Identity Federation is configured for SSL, the Oracle WebLogic Server Administration Server where the Fusion Middleware Control console is running should trust the SSL certificate (or the CA of the certificate) of the Oracle WebLogic Managed Server Managed Server where Oracle Identity Federation is running.

To configure the Oracle WebLogic Server to trust an SSL certificate (or its CA), you must add the necessary certificate as a trusted certificate entry to the JSSE Trust Keystore used by the Administration Server for SSL connections.

See Also:

- Oracle WebLogic Server documentation on how to change the JSSE Trust Keystore for use in SSL connections, if needed.
- Java documentation on how to use `keytool` to add a certificate as a Trusted Cert Entry in the JSSE Trust Keystore.

C.1.1.7 ParseException Message in Diagnostic Log

After installation, a configuration assistant performs a number of configuration updates to the Oracle Identity Federation server using MBeans. Another task periodically checks to see if the configuration files were changed so that the server can be notified.

A parsing error during this procedure can result in the following type of message in the diagnostic log file:

```
$DOMAIN_HOME/servers/wls_oif1/logs/wls_oif1-diagnostic.log
.
[org.xml.sax.SAXParseException: XML document structures must start and end
within the same entity.]
at
javax.xml.bind.helpers.AbstractUnmarshallerImpl.createUnmarshalExcept
ion(AbstractUnmarshallerImpl.java:315)
at
com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallerImpl.createUnmar
shalException(UnmarshallerImpl.java:514)
at
com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallerImpl.unmarshal0(
UnmarshallerImpl.java:215)
at
com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallerImpl.unmarshal(U
nmarshallerImpl.java:184)
at
javax.xml.bind.helpers.AbstractUnmarshallerImpl.unmarshal(AbstractUnm
arshallerImpl.java:137)
at
javax.xml.bind.helpers.AbstractUnmarshallerImpl.unmarshal(AbstractUnm
arshallerImpl.java:184)
at
oracle.as.config.persistence.jaxb.JAXBXmlPersistenceManagerImpl.load(
JAXBXmlPersistenceManagerImpl.java:156)
... 10 more
Caused by: org.xml.sax.SAXParseException: XML document structures must start
and
 end within the same entity.
at
com.sun.org.apache.xerces.internal.util.ErrorHandlerWrapper.createSAX
ParseException(ErrorHandlerWrapper.java:195)
at
com.sun.org.apache.xerces.internal.util.ErrorHandlerWrapper.fatalErro
r(ErrorHandlerWrapper.java:174)
.
```

Provided that the Oracle Identity Federation server is up and running (`/fed/idp/metadata` can be accessed without any errors), the message is harmless and has no effect on the stability of the server. The configuration change occurs as intended, and all the servers are notified of the change.

C.1.2 Oracle Identity Federation Configuration Issues

This section describes federation server configuration issues:

- Assertions Using SAML 1.x POST Method Fail in Japanese Locale
- Failed to find orclfednamevalue Error
- Configuring Audit Policies for Oracle Identity Federation Events
- Empty JNDI Name Message
- Database Column Too Short error for IDPPROVIDEDNAMEIDVALUE

C.1.2.1 Assertions Using SAML 1.x POST Method Fail in Japanese Locale

Problem

In the Japanese locale, assertions using the SAML 1.x POST method fail with this error:

```
ERROR: The SAML Response was not signed by the expected authority (RVE013)
```

The problem is due to the translated strings for OU and ST in the Signing Certificate Subject DN and the Signing Certificate Issuer DN.

Solution

As a workaround to this problem, the OU and ST values need to be replaced with the equivalent English strings. You can obtain the English value of the strings from the Issuer and Subject DN in the MyDomain configuration.

C.1.2.2 Failed to find orclfednamevalue Error

Problem

A schema violation error occurs when performing a Liberty 1.x / SAML 2.0 single sign-on operation, with the federation data store residing in an LDAP server. The `DOMAIN_HOME/servers/`*servername*`/logs/`*servername*`_diagnostics.log` shows the following error message:

```
javax.naming.directory.SchemaViolationException:
[LDAP: error code 65 -
Failed to find orclfednamevalue in mandatory or optional attribute list.]
```

This problem is seen if the schema of the federation data store's LDAP server has not been upgraded to include the Oracle Identity Federation attributes and object classes.

Solution

Upgrade the LDAP schema either at installation time (with the Advanced Installation mode), or after installation.

Upgrade Schema at Installation

To perform the upgrade at installation time, take these steps:

1. Choose the Advanced Installation mode.
2. On the "Select Oracle Identity Federation Advanced Flow Attributes" page, select the "LDAP" option from dropdown for Federation Store. This indicates that the federation records will be stored in an LDAP server whose schema must be upgraded.
3. On the "Specify LDAP Attributes for Federation Data Store" page, enter the LDAP connection information. The schema will then be upgraded as part of the installation process.

Post-Installation Schema Upgrade

To perform the upgrade post-installation, note that the Oracle Identity Federation installation includes `LDIF` files that you can execute using the `ldapmodify` tool to upgrade the schema of an LDAP server.

The `LDIF` file to use depends on the type of LDAP server used:

- `$Oracle_Home/fed/setup/ldap/userFedSchemaOid.ldif` if you use Oracle Internet Directory
- `$Oracle_Home/fed/setup/ldap/userFedSchemaSunOne5.ldif` if you use the Sun One Directory Server 5.x
- `$Oracle_Home/fed/setup/ldap/userFedSchemaSunOne6.ldif` if you use the Sun One Directory Server 6.x
- `$Oracle_Home/fed/setup/ldap/userFedSchemaTivoli.ldif` if you use IBM Tivoli
- `$Oracle_Home/fed/setup/ldap/userFedSchemaAD.ldif` if you use Microsoft Active Directory Server. In this case, you need to edit the LDIF file to replace the string `%DOMAIN_DN%` with your active directory domain suffix.

  An example suffix is `dc=mydomain,dc=mycompany,dc=com`.

Using `ldapmodify`, you can upgrade the LDAP schema with the LDIF file. For example:

```
ldapmodify -c -D BIND_DN_USERNAME
-w PASSWORD
-f $Oracle_Home/fed/setup/ldap/userFedSchemaOid.ldif
-h LDAP_HOSTNAME
-p LDAP_PORT -x
```

C.1.2.3 Configuring Audit Policies for Oracle Identity Federation Events

In Fusion Middleware Control, when you configure audit policies for Oracle Identity Federation events, note that even though the EM audit policy page shows both success and failure event types for all of the Oracle Identity Federation events, as a rule most Oracle Identity Federation configuration change events only have SUCCESS status by design.

The following events are the only Oracle Identity Federation events where failures are audited:

- encryptData
- decryptData
- verifySignature
- createSignature
- localAuthentication
- localLogout
- createActiveUserFederation
- deleteActiveUserFederation
- createUserFederation
- deleteUserFederation
- createUserSession

- deleteUserSession
- incomingMessage
- updateUserFederation

C.1.2.4 Empty JNDI Name Message

When using the Oracle Enterprise Manager Fusion Middleware Control to migrate a configuration data store to RDBMS, a JNDI name must be supplied. You may see the following warning message in the server log during the operation:

```
<Warning>
<oracle.security.fed.controller.web.servlet.OIFApplicationLifeCycleCallBack>
<FED-10205> <The JNDI name for RDBMS configuration backend is empty.>
 <Warning>
<oracle.security.fed.controller.web.servlet.OIFApplicationLifeCycleCallBack>
<FED-10205> <The JNDI name for RDBMS configuration backend is empty.>
```

This message can be ignored; however if it persists, check that the correct JNDI name is entered in Fusion Middleware Control.

C.1.2.5 Database Column Too Short error for IDPPROVIDEDNAMEIDVALUE

Problem

When Oracle Identity Federation is configured to use a database store for session and message data store, the following error is seen if data for `IDPPROVIDEDNAMEID` is over 200 characters long:

```
ORA-12899: value too large for column
"WDO_OIF"."ORAFEDTMPPROVIDERFED"."IDPPROVIDEDNAMEIDVALUE" (actual: 240,
maximum: 200)\n]
```

Solution

As a workaround to this problem, alter table `ORAFEDTMPPROVIDERFED` to increase the column size for "idpProvidedNameIDValue" to 240.

C.1.3 Oracle Single Sign-On Login Issues

This section describes issues that you may encounter when Oracle Identity Federation is the service provider (SP), and Oracle Single Sign-On is configured as the identity provider (IdP) at the back end. It contains these topics:

- Incorrect Login Page Appears
- Bookmarked Login Pages
- Unable to Modify File Used to Upload Provider Metadata

C.1.3.1 Incorrect Login Page Appears

Problem

The following setup produces an incorrect login page:

1. Oracle Identity Federation is configured as a service provider.
2. The partner application is configured to be protected by Oracle Identity Federation.
3. When a user tries to access the protected resource, the Oracle Single Sign-On login page appears instead of the intended Oracle Identity Federation login page.

This problem can occur if the partner application is incorrectly configured for `mod_osso`, causing the user to be prompted for local authentication.

Solution

The steps required to ensure that the partner application is correctly configured for `mod_osso` are outlined here.

1. Shut down Oracle HTTP Server and OC4J_SECURITY.
2. Edit the Oracle HTTP Server configuration file, `ORACLE_HOME/Apache/Apache/conf/httpd.conf`, located in the Oracle Application Server Infrastructure directory:
   - Add the osso_module to the server's loaded modules using the `AddModule` (Windows) and `LoadModule` (Windows, Linux) directives.
   - Add a virtual host to create a new partner application listener that will be protected by `mod_osso`.
3. Run `ssoreg` to manually configure `mod_osso` and the Oracle Single Sign-On server.

   Note:

   - Check that the `osso_APPLICATION_NAME.conf` matches the value defined in the virtual host configuration.

4. Restart Oracle HTTP Server (OHS) And `OC4J_SECURITY`.

C.1.3.2 Bookmarked Login Pages

Problem

Attempting to log in by means of a bookmarked login page returns an error. This is seen when a user follows this sequence:

- Perform a single sign-on (SSO) operation using SAML 2.0, Liberty, or WS-Federation protocols.
- On the login page, bookmark the page.
- Open a new browser instance and go to the bookmarked login page. Log in with valid user credentials.

The user will receive an error and SSO will fail.

Solution

Do not bookmark the login page. Oracle Identity Federation does not support the use of bookmarked login pages.

C.1.3.3 Unable to Modify File Used to Upload Provider Metadata

After using the Firefox browser to upload provider metadata from the Federations page in Fusion Middleware Control, you cannot modify the provider metadata file that you just uploaded until you restart Firefox.

The reason for this is the Live HTTP Header add-on in Firefox. Once you disable this add-on and restart Firefox, you can modify the provider metadata file after you have uploaded the metadata on the Federations page.

C.1.4 Oracle Access Manager Configuration Issues

This section describes issues that you may encounter when configuring Oracle Access Manager components at the back end. It contains these topics:

- AccessGate Permission Error
- Non-ASCII AccessGate ID
- Setting LD_ASSUME_KERNEL Value
- Using the Same Cookie Domain for Two Back-ends
- Oracle Access Manager Integration Issues

C.1.4.1 AccessGate Permission Error

Problem

The following setup produces an AccessGate configuration error:

1. The Access Server SDK specifies a certain user under whom the AccessGate runs.
2. Oracle Identity Federation is installed on the Linux or Solaris platform under a different user.
3. When you apply the AccessGate configuration page for the Oracle Access Manager user data store, you receive the error:

   ```
   AccessGate configuration failed. Reason: Preparing to connect to Access Server.
   Please wait. Error: Permission denied.
   ```

This error results from having different owners for the Oracle Identity Federation and Access Server SDK installations.

Solution

When Oracle Identity Federation is installed on Linux or Solaris, ensure that the AccessServerSDK files have the same owner and group as the Oracle Identity Federation installation.

C.1.4.2 Non-ASCII AccessGate ID

Problem

If you attempt to configure an Oracle Access Manager User Data Store with an AccessGate, and the AccessGate ID contains non-Latin characters (for example, "ÆÖ2"), you get the error:

```
AccessGate configuration failed. Reason: Preparing to connect to Access
Server. Please wait. Client authentication failed, please verify your
AccessGate ID.
```

The problem also occurs when the AccessGate ID contains non-ASCII Latin-1 characters (for example, "Ådmïn").

Solution

Use only ASCII characters in the AccessGate ID for the Oracle Identity Federation AccessGate.

C.1.4.3 Setting LD_ASSUME_KERNEL Value

Problem

Oracle Identity Federation fails to configure Oracle Access Manager for use as SP integration module when you try to set up this configuration using Fusion Middleware Control. This occurs when operating with an Oracle Access Manager back-end version 10.1.4.2 or earlier.

This problem may be due to an incorrect setting for the `LD_ASSUME_KERNEL` environment variable. This variable must be set to `2.4.19`, because the Access Server SDK supports the Linux threading model and not the native posix thread library (NPTL).

For example, if `LD_ASSUME_KERNEL` is not set, you may see this type of error in the Oracle Identity Federation log files:

```
com.oblix.access.ObAccessException: Env variable LD_ASSUME_KERNEL not set to
2.4.19.
      at com.oblix.access.ObConfig.jni_initialize(Native Method)
      ...
```

The browser also shows an error during this setup (that is, when you try to configure Oracle Access Manager as an SP integration module using Fusion Middleware Control).

Solution

Refer to Section 3.2.4.3, "Integrate Oracle Access Manager as an SP Integration Module" which describes how to set `LD_ASSUME_KERNEL` in the Oracle WebLogic Server environment.

C.1.4.4 Using the Same Cookie Domain for Two Back-ends

Problem

If your configuration involves two providers with Oracle Access Manager back-ends (IdP and SP2, for example), and both instances are using the same cookie domain, you may see an error when attempting single sign-on.

This problem occurs when multiple Oracle Access Manager providers are using the same cookie domain.

Solution

You can resolve the issue by changing the Oracle Access Manager instances to use different cookie domains, or by using a different back-end (such as an LDAP back-end) at the IdP.

C.1.4.5 Oracle Access Manager Integration Issues

Problem

You may see one of these errors when configuring Oracle Identity Federation to integrate with Oracle Access Manager 10*g* through Fusion Middleware Control:

```
2009-03-09T21:54:56.354-07:00] [wls_oif1] [INCIDENT_ERROR] [FED-10192]
[oracle.security.fed.admin.config.mbeans.OAMConfigUtils] [tid:
[ACTIVE].ExecuteThread: '3' for queue: 'weblogic.kernel.Default (self-tuning)']
[userId: <WLS Kernel>] [ecid: 0000HziivsZ7IBT6uBr2EH19hQcU000017,1:5006] [APP:
Oracle Identity Federation#11.1.1.1.0] Oracle Access Server SDK could not be
initialized
```

Solution

Check the configuration of Oracle Identity Federation with Oracle Access Manager and AccessServerSDK.

Problem

You may see this error when Oracle Identity Federation is integrated with Oracle Access Manager 10*g* and a user attempts to access the OAM-protected resource:

```
Error:  oracle.security.fed.event.EventException:
com.oblix.access.ObAccessException:
Unprotected resource GET used in an ObAuthenticationScheme or ObUserSession
constructor
```

Solution

If this error is encountered, check that the host ID has the Oracle Identity Federation port (for example, port 7499).

C.1.5 Operating System Configuration Issues

This section describes issues related to the configuration of the operating system of the machine where Oracle Identity Federation is installed:

- File Descriptors on Linux

Note: Some issues listed in this section may have a system-wide impact on the Oracle Identity Federation server, while others may only impact a specific component such as a particular federation partner.

C.1.5.1 File Descriptors on Linux

Problem

You may experience intermittent Oracle Identity Federation server crashes with this error message in the log file:

```
java.net.SocketException: Too many open files
```

This error occurs when the file descriptor limit is reached.

Solution

Increase the file descriptor limit, which is specified in the `/etc/security/limits.conf` configuration file.

Note: If no file descriptor limit is defined, the server uses a default value of 1024.

In this example, the file descriptor limit is being set to a value of 16K:

```
soft nofile 16384
hard nofile 16384
```

Reboot the machine after changing the value.

C.1.6 Runtime/Single Sign-On Issues

This section describes runtime and single sign-on issues that you may encounter when using Oracle Identity Federation:

- Bookmarking a WS-Federation Protected Resource

- SP Unable to Map NameID to Local User

C.1.6.1 Bookmarking a WS-Federation Protected Resource

Problem

If a user bookmarks a WS-Federation protected resource, and the service provider is using an Oracle Single Sign-On back-end, the user will receive an error when later trying to access the bookmark.

Solution

Accessing WS-Federation protected resources directly is not supported if the SP is using an Oracle Single Sign-On back-end. Users should not bookmark and later attempt to access the resource in this scenario.

C.1.6.2 SP Unable to Map NameID to Local User

Problem

When using a Name ID format such as email address (mapped to an attribute value such as 'mail'), the user is initially able to perform SSO, and federation records are generated at the IdP and SP.

However, it is possible that the value for the attribute (such as mail) may change. After the change, the IdP still sends the old Name ID value (in this case, the old email address) in the assertion, causing SSO to fail.

Solution

The Oracle Identity Federation administrator should update the federation records to reflect the new attribute values.

C.1.7 Performance Issues

This section contains these topics:

- Internal Error 500 when Using LDAP Store

C.1.7.1 Internal Error 500 when Using LDAP Store

Problem

When using an LDAP-based data store, users may see the following error in the log when there is heavy login activity at the server:

```
Internal Error: 500
```

Solution

This is likely due to inadequate tuning of the LDAP store. Make sure that the connection pool for your LDAP server is properly tuned.

Glossary

3DES

See Triple Data Encryption Standard (3DES).

account lockout

A security feature that locks a user account if repeated failed logon attempts occur within a specified amount of time, based on security policy settings. Account lockout occurs in Oracle Single Sign-On when a user submits an account and password combination from any number of workstations more times than is permitted by Oracle Internet Directory. The default lockout period is 24 hours.

Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is a symmetric cryptography algorithm that is intended to replace Data Encryption Standard (DES). AES is a Federal Information Processing Standard (FIPS) for the encryption of commercial and government data.

advanced symmetric replication (ASR)

See Oracle Database Advanced Replication.

AES

See Advanced Encryption Standard (AES).

anonymous authentication

The process by which a directory authenticates a user without requiring a user name and password combination. Each anonymous user then exercises the privileges specified for anonymous users.

API

See application programming interface (API).

application programming interface (API)

A series of software routines and development tools that comprise an interface between a computer application and lower-level services and functions (such as the operating system, device drivers, and other software applications). APIs serve as building blocks for programmers putting together software applications. For example, LDAP-enabled clients access Oracle Internet Directory information through programmatic calls available in the LDAP API.

application service provider

Application Service Providers (ASPs) are third-party entities that manage and distribute software-based services and solutions to customers across a wide area network from a central data center. In essence, ASPs are a way for companies to outsource some or almost all aspects of their information technology needs.

artifact profile

An authentication mechanism which transmits data using a compact reference to an assertion, called an artifact, instead of sending the full assertion. This profile accommodates browsers which handle a limited number of characters.

ASN.1

Abstract Syntax Notation One (ASN.1) is an International Telecommunication Union (ITU) notation used to define the syntax of information data. ASN.1 is used to describe structured information, typically information that is to be conveyed across some communications medium. It is widely used in the specification of Internet protocols.

assertion

An assertion is a statement used by providers in security domains to exchange information about a subject seeking access to a resource. Identity providers and service providers exchange assertions about identities to make authentication and authorization decisions, and to determine and enforce security policies protecting the resource.

asymmetric algorithm

A cryptographic algorithm that uses different keys for encryption and decryption.

See also: public key cryptography.

asymmetric cryptography

See public key cryptography.

authentication

The process of verifying the identity claimed by an entity based on its credentials. Authentication of a user is generally based on something the user knows or has (for example, a password or a certificate).

Authentication of an electronic message involves the use of some kind of system (such as public key cryptography) to ensure that a file or message which claims to originate from a given individual or company actually does, and a check based on the contents of a message to ensure that it was not modified in transit.

authentication level

An Oracle Single Sign-On parameter that enables you to specify a particular authentication behavior for an application. You can link this parameter with a specific authentication plugin.

authentication plugin

An implementation of a specific authentication method. Oracle Single Sign-On has Java plug-ins for password authentication, digital certificates, Windows native authentication, and third-party access management.

authorization

The process of granting or denying access to a service or network resource. Most security systems are based on a two step process. The first stage is authentication, in which a user proves his or her identity. The second stage is authorization, in which a user is allowed to access various resources based on his or her identity and the defined authorization policy.

authorization policy

Authorization policy describes how access to a protected resource is governed. Policy maps identities and objects to collections of rights according to some system model. For example, a particular authorization policy might state that users can access a sales report only if they belong to the sales group.

basic authentication

An authentication protocol supported by most browsers in which a Web server authenticates an entity with an encoded user name and password passed via data transmissions. Basic authentication is sometimes called plaintext authentication because the base-64 encoding can be decoded by anyone with a freely available decoding utility. Note that encoding is not the same as encryption.

Basic Encoding Rules (BER)

Basic Encoding Rules (BER) are the standard rules for encoding data units set forth in ASN.1. BER is sometimes incorrectly paired with ASN.1, which applies only to the abstract syntax description language, not the encoding technique.

BER

See Basic Encoding Rules (BER).

binding

In networking, binding is the establishment of a logical connection between communicating entities.

In the case of Oracle Internet Directory, binding refers to the process of authenticating to the directory.

The formal set of rules for carrying a SOAP message within or on top of another protocol (underlying protocol) for the purpose of exchange is also called a binding.

CA

See Certificate Authority (CA).

CA certificate

A Certificate Authority (CA) signs all certificates that it issues with its private key. The corresponding Certificate Authority's public key is itself contained within a certificate, called a CA Certificate (also referred to as a root certificate). A browser must contain the CA Certificate in its list of trusted root certificates in order to trust messages signed by the CA's private key.

cache

Generally refers to an amount of quickly accessible memory in your computer. However, on the Web it more commonly refers to where the browser stores downloaded files and graphics on the user's computer.

CBC

See cipher block chaining (CBC).

certificate

A certificate is a specially formatted data structure that associates a public key with the identity of its owner. A certificate is issued by a Certificate Authority (CA). It contains the name, serial number, expiration dates, and public key of a particular entity. The certificate is digitally signed by the issuing CA so that a recipient can verify that the certificate is real. Most digital certificates conform to the X.509 standard.

Certificate Authority (CA)

A Certificate Authority (CA) is a trusted third party that issues, renews, and revokes digital certificates. The CA essentially vouches for a entity's identity, and may delegate the verification of an applicant to a Registration Authority (RA). Some well known Certificate Authorities (CAs) include Digital Signature Trust, Thawte, and VeriSign.

certificate chain

An ordered list of certificates containing one or more pairs of a user certificate and its associated CA certificate.

certificate revocation list (CRL)

A Certificate Revocation List (CRL) is a list of digital certificates which have been revoked by the Certificate Authority (CA) that issued them.

change logs

A database that records changes made to a directory server.

cipher

See cryptographic algorithm.

cipher block chaining (CBC)

Cipher block chaining (CBC) is a mode of operation for a block cipher. CBC uses what is known as an initialization vector (IV) of a certain length. One of its key characteristics is that it uses a chaining mechanism that causes the decryption of a block of ciphertext to depend on all the preceding ciphertext blocks. As a result, the entire validity of all preceding blocks is contained in the immediately previous ciphertext block.

cipher suite

In Secure Sockets Layer (SSL), a set of authentication, encryption, and data integrity algorithms used for exchanging messages between network nodes. During an SSL handshake, the two nodes negotiate to see which cipher suite they will use when transmitting messages back and forth.

ciphertext

Ciphertext is the result of applying a cryptographic algorithm to readable data (plaintext) in order to render the data unreadable by all entities except those in possession of the appropriate key.

circle of trust

A trust relationship among a set of identity providers and service providers that allows a Principal to use a single federated identity and single sign-on when conducting business transactions with providers within that set.

Businesses federate or affiliate together into circles of trust based on Liberty-enabled technology and on operational agreements that define trust relationships between the businesses.

See also: federation, Liberty Alliance.

claim

A claim is a declaration made by an entity (for example, a name, identity, key, group, and so on).

client SSL certificates

A type of certificate used to identify a client machine to a server through Secure Sockets Layer (SSL) (client authentication).

cluster

A collection of interconnected usable whole computers that is used as a single computing resource. Hardware clusters provide high availability and scalability.

CMP

See certificate management protocol (CMP).

CMS

See Cryptographic Message Syntax (CMS).

code signing certificates

A type of certificate used to identify the entity who signed a Java program, Java Script, or other signed file.

Cold Failover Cluster

Oracle Application Server Cold Failover Clusters are a high availability solution where the Oracle Application Server Infrastructure is typically deployed on a two-node hardware cluster with a shared storage device. Once node is active or "hot," meaning it is running the Infrastructure, while the other node is "cold" and is not running the Infrastructure. When the active node fails, the clusterware switches Infrastructure operations to the previously "cold" node and the Infrastructure is started on that node.

concurrency

The ability to handle multiple requests simultaneously. Threads and processes are examples of concurrency mechanisms.

confidentiality

In cryptography, confidentiality (also known as privacy) is the ability to prevent unauthorized entities from reading data. This is typically achieved through encryption.

connect descriptor

A specially formatted description of the destination for a network connection. A connect descriptor contains destination service and network route information.

The destination service is indicated by using its service name for the Oracle Database or its Oracle System Identifier (SID) for Oracle release 8.0 or version 7 databases. The network route provides, at a minimum, the location of the listener through use of a network address.

contention

Competition for resources.

context prefix

The distinguished name (DN) of the root of a naming context.

CRL

See certificate revocation list (CRL).

CRMF

See certificate request message format (CRMF).

cryptographic algorithm

A cryptographic algorithm is a defined sequence of processes to convert readable data (plaintext) to unreadable data (ciphertext) and vice versa. These conversions require some secret knowledge, normally contained in a key. Examples of cryptographic algorithms include DES, AES, Blowfish, and RSA.

Cryptographic Message Syntax (CMS)

Cryptographic Message Syntax (CMS) is a syntax defined in RFC 3369 for signing, digesting, authenticating, and encrypting digital messages.

cryptography

The process of protecting information by transforming it into an unreadable format. The information is encrypted using a key, which makes the data unreadable, and is then decrypted later when the information needs to be used again. See also public key cryptography and symmetric cryptography.

dads.conf

A configuration file for Oracle HTTP Server that is used to configure a database access descriptor (DAD).

DAS

See Oracle Delegated Administration Services. (DAS).

Data Encryption Standard (DES)

Data Encryption Standard (DES) is a widely used symmetric cryptography algorithm developed in 1974 by IBM. It applies a 56-bit key to each 64-bit block of data. DES and 3DES are typically used as encryption algorithms by S/MIME.

data integrity

The guarantee that the contents of the message received were not altered from the contents of the original message sent.

See also: integrity.

database access descriptor (DAD)

Database connection information for a particular Oracle Application Server component, such as the Oracle Single Sign-On schema.

decryption

The process of converting the contents of an encrypted message (ciphertext) back into its original readable format (plaintext).

defederation

The act of unlinking a user's account from an identity provider or service provider.

DER

See Distinguished Encoding Rules (DER).

DES

See Data Encryption Standard (DES).

DIB

See directory information base (DIB).

Diffie-Hellman

Diffie-Hellman (DH) is a public key cryptography protocol that allows two parties to establish a shared secret over an unsecure communications channel. First published in 1976, it was the first workable public key cryptographic system.

See also: symmetric algorithm.

digest

See message digest.

digital certificate

See certificate.

digital signature

A digital signature is the result of a two-step process applied to a given block of data. First, a hash function is applied to the data to obtain a result. Second, that result is encrypted using the signer's private key. Digital signatures can be used to ensure integrity, message authentication, and non-repudiation of data. Examples of digital signature algorithms include DSA, RSA, and ECDSA.

Digital Signature Algorithm (DSA)

The Digital Signature Algorithm (DSA) is an asymmetric algorithm that is used as part of the Digital Signature Standard (DSS). It cannot be used for encryption, only for digital signatures. The algorithm produces a pair of large numbers that enable the authentication of the signatory, and consequently, the integrity of the data attached. DSA is used both in generating and verifying digital signatures.

See also: Elliptic Curve Digital Signature Algorithm (ECDSA).

Distinguished Encoding Rules (DER)

Distinguished Encoding Rules (DER) are a set of rules for encoding ASN.1 objects in byte-sequences. DER is a special case of Basic Encoding Rules (BER).

distinguished name (DN)

A X.500 distinguished name (DN) is a unique name for a node in a directory tree. A DN is used to provide a unique name for a person or any other directory entry. A DN is a concatenation of selected attributes from each node in the tree along the path from the root node to the named entry's node. For example, in LDAP notation, the DN for a person named John Smith working at Oracle's US office would be: "cn=John Smith, ou=People, o=Oracle, c=us".

DN

See distinguished name (DN).

Document Type Definition (DTD)

A Document Type Definition (DTD) is a document that specifies constraints on the tags and tag sequences that are valid for a given XML document. DTDs follow the rules of Simple Generalized Markup Language (SGML), the parent language of XML.

domain

A domain is a Web site and applications that enable a principal to utilize resources. In federated identity management (FIM), a federated site acts as an identity provider (also known as the source domain), a service provider (or destination domain), or both.

DSA

See Digital Signature Algorithm (DSA) or directory system agent (DSA).

DSE

See directory-specific entry (DSE).

DTD

See Document Type Definition (DTD).

ECC

See Elliptic Curve Cryptography (ECC).

ECDSA

See Elliptic Curve Digital Signature Algorithm (ECDSA).

EJB

See Enterprise Java Bean (EJB).

Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is an alternative to the RSA encryption system which is based on the difficulty of solving elliptic curve discrete logarithm problems rather than on factoring large numbers. Developed and marketed by Certicom, ECC is especially suitable for environments, such as wireless devices and PC cards, where computational power is limited and high speed is required. For any given key size (measured in bits) ECC provides more security (is harder to decrypt without the key) than RSA.

Elliptic Curve Digital Signature Algorithm (ECDSA)

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analog of the Digital Signature Algorithm (DSA) standard. The advantages of ECDSA compared to RSA-like schemes are shorter key lengths and faster signing and decryption. For example, a 160 (210) bit ECC key is expected to give the same security as a 1024 (2048) bit RSA key, and the advantage increases as level of security is raised.

encryption

Encryption is the process of converting plaintext to ciphertext by applying a cryptographic algorithm.

encryption certificate

An encryption certificate is a certificate containing a public key that is used to encrypt electronic messages, files, documents, or data transmission, or to establish or exchange a session key for these same purposes.

end-to-end security

This is a property of message-level security that is established when a message traverses multiple applications within and between business entities and is secure over its full route through and between the business entities.

Enterprise Java Bean (EJB)

Enterprise JavaBeans (EJBs) are a Java API developed by Sun Microsystems that defines a component architecture for multi-tier client/server systems. Because EJB systems are written in Java, they are platform independent. Being object oriented, they can be implemented into existing systems with little or no recompiling and configuring.

Enterprise Manager

See Oracle Enterprise Manager.

failover

The process of failure recognition and recovery. In an Oracle Application Server Cold Failover Cluster (Identity Management), an application running on one cluster node is transparently migrated to another cluster node. During this migration, clients accessing the service on the cluster see a momentary outage and may need to reconnect once the failover is complete.

Federal Information Processing Standards (FIPS)

Federal Information Processing Standards (FIPS) are standards for information processing issued by the US government Department of Commerce's National Institute of Standards and Technology (NIST).

federated identity management (FIM)

The agreements, standards, and technologies that make identity and entitlements portable across autonomous domains. FIM makes it possible for an authenticated user to be recognized and take part in personalized services across multiple domains. It avoids pitfalls of centralized storage of personal information, while allowing users to link identity information between different accounts. Federated identity requires two key components: trust and standards. The trust model of federated identity management is based on circle of trust. The standards are defined by the Liberty Alliance Project.

federation

See identity federation.

filter

A filter is an expression that defines the entries to be returned from a request or search on a directory. Filters are typically expressed as DNs, for example: `cn=susie smith,o=acme,c=us`.

FIM

See federated identity management (FIM).

FIPS

See Federal Information Processing Standards (FIPS).

forced authentication

The act of forcing a user to reauthenticate if he or she has been idle for a preconfigured amount of time. Oracle Single Sign-On enables you to specify a global user inactivity timeout. This feature is intended for installations that have sensitive applications.

GET

An authentication method whereby login credentials are submitted as part of the login URL.

global administrator

In a hosted environment, one enterprise—for example, an application service provider—makes Oracle components available to multiple other enterprises and stores information for them. In such an environment, a global administrator performs activities that span the entire directory.

global unique identifier (GUID)

An identifier generated by the system and inserted into an entry when the entry is added to the directory. In a multimaster replicated environment, the GUID, not the DN, uniquely identifies an entry. The GUID of an entry cannot be modified by a user.

global user inactivity timeout

An optional feature of Oracle Single Sign-On that forces users to reauthenticate if they have been idle for a preconfigured amount of time. The global user inactivity timeout is much shorter than the single sign-out session timeout.

globally unique user ID

A numeric string that uniquely identifies a user. A person may change or add user names, passwords, and distinguished names, but her globally unique user ID always remains the same.

grace login

A login occurring within the specified period before password expiration.

guest user

One who is not an anonymous user, and, at the same time, does not have a specific user entry.

GUID

See global unique identifier (GUID).

handshake

A protocol two computers use to initiate a communication session.

hash

A number generated from a string of text with an algorithm. The hash value is substantially smaller than the text itself. Hash numbers are used for security and for faster access to data.

See also: hash function.

hash function

In cryptography, a hash function or one-way hash function is an algorithm that produces a given value when applied to a given block of data. The result of a hash function can be used to ensure the integrity of a given block of data. For a hash function to be considered secure, it must be very difficult, given a known data block and a known result, to produce another data block that produces the same result.

Hashed Message Authentication Code (HMAC)

Hashed Message Authentication Code (HMAC) is a hash function technique used to create a secret hash function output. This strengthens existing hash functions such as MD5 and SHA. It is used in transport layer security (TLS).

HMAC

See Hashed Message Authentication Code (HMAC).

HTTP

The Hyper Text Transfer Protocol (HTTP) is the protocol used between a Web browser and a server to request a document and transfer its contents. The specification is maintained and developed by the World Wide Web Consortium.

HTTP Server

See Oracle HTTP Server.

httpd.conf

The file used to configure Oracle HTTP Server.

iASAdmins

The administrative group responsible for user and group management functions in Oracle Application Server. The Oracle Single Sign-On administrator is a member of the group iASAdmins.

identity federation

The linking of two or more accounts a Principal may hold with one or more identity providers or service providers within a given circle of trust.

When users federate the otherwise isolated accounts they have with businesses, known as their local identities, they create a relationship between two entities, an association comprising any number of identity providers and service providers.

identity management

The process by which the complete security lifecycle for network entities is managed in an organization. It typically refers to the management of an organization's application users, where steps in the security life cycle include account creation, suspension, privilege modification, and account deletion. The network entities managed may also include devices, processes, applications, or anything else that needs to interact in a networked environment. Entities managed by an identity management process may also include users outside of the organization, for example customers, trading partners, or Web services.

identity management infrastructure database

The database that contains data for Oracle Single Sign-On and Oracle Internet Directory.

identity provider

One of the three primary roles defined in the identity federation protocols supported by Oracle Identity Federation. (The others are service provider and Principal.) The identity provider is responsible for managing and authenticating a set of identities within a given circle of trust.

A service provider (relying party in SAML), in turn, provides services or goods to a principal based on the identity provider's authentication of a principal's identity.

Identity providers are service providers offering business incentives so that other service providers affiliate with them. An identity provider will typically authenticate and asserts a principal's identity.

IdMBridge

The IdMBridge binds and provides user attributes for assertions, and is responsible for communication with various authoritative sources of data.

import agent

In an Oracle Directory Integration and Provisioning environment, an agent that imports data into Oracle Internet Directory.

import data file

In an Oracle Directory Integration and Provisioning environment, the file containing the data imported by an import agent.

infrastructure tier

The Oracle Application Server components responsible for identity management. These components are Oracle Single Sign-On, Oracle Delegated Administration Services, and Oracle Internet Directory.

inherit

When an object class has been derived from another class, it also derives, or inherits, many of the characteristics of that other class. Similarly, an attribute subtype inherits the characteristics of its supertype.

integrity

In cryptography, integrity is the ability to detect if data has been modified by entities that are not authorized to modify it.

Internet Directory

See Oracle Internet Directory.

Internet Engineering Task Force (IETF)

The principal body engaged in the development of new Internet standard specifications. It is an international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

Internet Message Access Protocol (IMAP)

A protocol allowing a client to access and manipulate electronic mail messages on a server. It permits manipulation of remote message folders, also called mailboxes, in a way that is functionally equivalent to local mailboxes.

J2EE

See Java 2 Platform, Enterprise Edition (J2EE).

Java 2 Platform, Enterprise Edition (J2EE)

Java 2 Platform, Enterprise Edition (J2EE) is an environment for developing and deploying enterprise applications, defined by Oracle. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, Web-based applications.

Java Server Page (JSP)

JavaServer Pages (JSP), a server-side technology, are an extension to the Java servlet technology that was developed by Oracle. JSPs have dynamic scripting capability that works in tandem with HTML code, separating the page logic from the static elements (the design and display of the page). Embedded in the HTML page, the Java source code and its extensions help make the HTML more functional, being used in dynamic database queries, for example.

JSP

See Java Server Page (JSP).

key

A key is a data structure that contains some secret knowledge necessary to successfully encrypt or decrypt a given block of data. The larger the key, the harder it is to crack a block of encrypted data. For example, a 256-bit key is more secure than a 128-bit key.

key pair

A public key and its associated private key.

See also: public/private key pair.

latency

The time a client has to wait for a given directory operation to complete. Latency can be defined as wasted time. In networking discussions, latency is defined as the travel time of a packet from source to destination.

LDAP

See Lightweight Directory Access Protocol (LDAP).

LDAP connection cache

To improve throughput, the Oracle Single Sign-On server caches and then reuses connections to Oracle Internet Directory.

LDAP Data Interchange Format (LDIF)

A common, text-based format for exchanging directory data between systems. The set of standards for formatting an input file for any of the LDAP command-line utilities.

LDIF

See LDAP Data Interchange Format (LDIF).

legacy application

An older application that cannot be modified to delegate authentication to the Oracle Single Sign-On server. Also known as an external application.

Liberty Alliance

The Liberty Alliance Project is a consortium of companies, non-profits, and non-government organizations around the globe. It is committed to developing an open standard for federated identity management (FIM) and identity-based web services supporting current and emerging network devices.

Lightweight Directory Access Protocol (LDAP)

A set of protocols for accessing information in directories. LDAP supports TCP/IP, which is necessary for any type of Internet access. Its framework of design conventions supports industry-standard directory products, such as Oracle Internet Directory. Because it is a simpler version of the X.500 standard, LDAP is sometimes called X.500 light.

load balancer

Hardware devices and software that balance connection requests between two or more servers, either due to heavy load or failover. BigIP, Alteon, or Local Director are all popular hardware devices. Oracle Web Cache is an example of load balancing software.

logical host

In an Oracle Application Server Cold Failover Cluster (Identity Management), one or more disk groups and pairs of host names and IP addresses. It is mapped to a physical host in the cluster. This physical host impersonates the host name and IP address of the logical host.

MAC

See message authentication code (MAC).

man-in-the-middle

A security attack characterized by the third-party, surreptitious interception of a message. The third-party, the *man-in-the-middle*, decrypts the message, re-encrypts it (with or without alteration of the original message), and retransmits it to the originally-intended recipient—all without the knowledge of the legitimate sender and receiver. This type of security attack works only in the absence of authentication.

mapping rules file

In an Oracle Directory Integration and Provisioning environment, the file that specifies mappings between Oracle Internet Directory attributes and those in a connected directory.

master definition site (MDS)

In replication, a master definition site is the Oracle Internet Directory database from which the administrator runs the configuration scripts.

master site

In replication, a master site is any site other than the master definition site (MDS) that participates in LDAP replication.

MD2

Message Digest Two (MD2) is a message digest hash function. The algorithm processes input text and creates a 128-bit message digest which is unique to the message and can be used to verify data integrity. MD2 was developed by Ron Rivest for RSA Security and is intended to be used in systems with limited memory, such as smart cards.

MD4

Message Digest Four (MD4) is similar to MD2 but designed specifically for fast processing in software.

MD5

Message Digest Five (MD5) is a message digest hash function. The algorithm processes input text and creates a 128-bit message digest which is unique to the message and can be used to verify data integrity. MD5 was developed by Ron Rivest after potential weaknesses were reported in MD4. MD5 is similar to MD4 but slower because more manipulation is made to the original data.

MDS

See master definition site (MDS).

message authentication

The process of verifying that a particular message came from a particular entity.

See also: authentication.

message authentication code (MAC)

The Message Authentication Code (MAC) is a result of a two-step process applied to a given block of data. First, the result of a hash function is obtained. Second, that result is encrypted using a secret key. The MAC can be used to authenticate the source of a given block of data.

message digest

The result of a hash function.

See also: hash.

metadirectory

A directory solution that shares information between all enterprise directories, integrating them into one virtual directory. It centralizes administration, thereby reducing administrative costs. It synchronizes data between directories, thereby ensuring that it is consistent and up-to-date across the enterprise.

middle tier

That portion of a Oracle Single Sign-On instance that consists of the Oracle HTTP Server and OC4J. The Oracle Single Sign-On middle tier is situated between the identity management infrastructure database and the client.

mod_osso

A module on the Oracle HTTP Server that enables applications protected by Oracle Single Sign-On to accept HTTP headers instead of a user name and password once the user has logged into the Oracle Single Sign-On server. The values for these headers are stored in the mod_osso cookie.

mod_osso cookie

User data stored on the HTTP server. The cookie is created when a user authenticates. When the same user requests another application, the Web server uses the information in the mod_osso cookie to log the user in to the application. This feature speeds server response time.

mod_proxy

A module on the Oracle HTTP Server that makes it possible to use mod_osso to enable single sign-on to legacy, or external applications.

MTS

See shared server.

multimaster replication

Also called peer-to-peer or *n*-way replication, a type of replication that enables multiple sites, acting as equals, to manage groups of replicated data. In a multimaster replication environment, each node is both a supplier and a consumer node, and the entire directory is replicated on each node.

naming attribute

The attribute used to compose the RDN of a new user entry created through Oracle Delegated Administration Services or Oracle Internet Directory Java APIs. The default value for this is `cn`.

nickname attribute

The attribute used to uniquely identify a user in the entire directory. The default value for this is `uid`. Applications use this to resolve a simple user name to the complete distinguished name. The user nickname attribute cannot be multi-valued—that is, a given user cannot have multiple nicknames stored under the same attribute name.

non-repudiation

In cryptography, the ability to prove that a given digital signature was produced with a given entity's private key, and that a message was sent untampered at a given point in time.

OASIS

Organization for the Advancement of Structured Information Standards. OASIS is a worldwide not-for-profit consortium that drives the development, convergence and adoption of e-business standards.

object class

In LDAP, object classes are used to group information. Typically an object class models a real-world object such as a person or a server. Each directory entry belongs to one or more object classes. The object class determines the attributes that make up an entry. One object class can be derived from another, thereby inheriting some of the characteristics of the other class.

OC4J

See Oracle Containers for J2EE (OC4J).

OCA

See Oracle Certificate Authority.

OCI

See Oracle Call Interface (OCI).

OCSP

See Online Certificate Status Protocol (OCSP).

OEM

See Oracle Enterprise Manager.

OID

See Oracle Internet Directory.

OID Control Utility

A command-line tool for issuing run-server and stop-server commands. The commands are interpreted and executed by the OID Monitor process.

OID Database Password Utility

The utility used to change the password with which Oracle Internet Directory connects to an Oracle Database.

OID Monitor

The Oracle Internet Directory component that initiates, monitors, and terminates the Oracle Internet Directory Server processes. It also controls the replication server if one is installed, and Oracle Directory Integration and Provisioning Server.

Online Certificate Status Protocol (OCSP)

Online Certificate Status Protocol (OCSP) is one of two common schemes for checking the validity of digital certificates. The other, older method, which OCSP has superseded in some scenarios, is certificate revocation list (CRL). OCSP is specified in RFC 2560.

one-way function

A function that is easy to compute in one direction but quite difficult to reverse compute, that is, to compute in the opposite direction.

one-way hash function

A one-way function that takes a variable sized input and creates a fixed size output.

See also: hash function.

Oracle Application Server Single Sign-On

Oracle Single Sign-On consists of program logic that enables you to log in securely to applications such as expense reports, mail, and benefits. These applications take two forms: partner applications and external applications. In both cases, you gain access to several applications by authenticating only once.

Oracle Call Interface (OCI)

An application programming interface (API) that enables you to create applications that use the native procedures or function calls of a third-generation language to access an Oracle Database server and control all phases of SQL statement execution.

Oracle Certificate Authority

Oracle Application Server Certificate Authority is a Certificate Authority (CA) for use within your Oracle Application Server environment. OracleAS Certificate Authority uses Oracle Internet Directory as the storage repository for certificates. OracleAS Certificate Authority integration with Oracle Single Sign-On and Oracle Internet Directory provides seamless certificate provisioning mechanisms for applications relying on them. A user provisioned in Oracle Internet Directory and authenticated in Oracle Single Sign-On can choose to request a digital certificate from OracleAS Certificate Authority.

Oracle CMS

Oracle CMS implements the IETF Cryptographic Message Syntax (CMS) protocol. CMS defines data protection schemes that allow for secure message envelopes.

Oracle Containers for J2EE (OC4J)

A lightweight, scalable container for Java 2 Platform, Enterprise Edition (J2EE).

Oracle Crypto

Oracle Crypto is a pure Java library that provides core cryptography algorithms.

Oracle Database Advanced Replication

A feature in the Oracle Database that enables database tables to be kept synchronized across two Oracle databases.

Oracle Delegated Administration Services

A set of individual, pre-defined services—called Oracle Delegated Administration Services units—for performing directory operations on behalf of a user. Oracle Internet Directory Self-Service Console makes it easier to develop and deploy administration solutions for both Oracle and third-party applications that use Oracle Internet Directory.

Oracle Directory Integration and Provisioning

A collection of interfaces and services for integrating multiple directories by using Oracle Internet Directory and several associated plug-ins and connectors. A feature of Oracle Internet Directory that enables an enterprise to use an external user repository to authenticate to Oracle products.

Oracle Directory Manager

A Java-based tool with a graphical user interface for administering Oracle Internet Directory.

Oracle Enterprise Manager

A separate Oracle product that combines a graphical console, agents, common services, and tools to provide an integrated and comprehensive systems management platform for managing Oracle products.

Oracle HTTP Server

Software that processes Web transactions that use the Hypertext Transfer Protocol (HTTP). Oracle uses HTTP software developed by the Apache Group.

Oracle Identity Management

An infrastructure enabling deployments to manage centrally and securely all enterprise identities and their access to various applications in the enterprise.

Oracle Internet Directory

A general purpose directory service that enables retrieval of information about dispersed users and network resources. It combines Lightweight Directory Access Protocol (LDAP) Version 3 with the high performance, scalability, robustness, and availability of the Oracle Database.

Oracle Liberty SDK

Oracle Liberty SDK implements the Liberty Alliance Project specifications enabling federated single sign-on between third-party Liberty-compliant applications.

Oracle Net Services

The foundation of the Oracle family of networking products, allowing services and their client applications to reside on different computers and communicate. The main function of Oracle Net Services is to establish network sessions and transfer data between a client application and a server. Oracle Net Services is located on each computer in the network. Once a network session is established, Oracle Net Services acts as a data courier for the client and the server.

Oracle PKI certificate usages

Defines Oracle application types that a certificate supports.

Oracle PKI SDK

Oracle PKI SDK implements the security protocols that are necessary within public key infrastructure (PKI) implementations.

Oracle SAML

Oracle SAML provides a framework for the exchange of security credentials among disparate systems and applications in an XML-based format as outlined in the OASIS specification for the Security Assertions Markup Language (SAML).

Oracle Security Engine

Oracle Security Engine extends Oracle Crypto by offering X.509 based certificate management functions. Oracle Security Engine is a superset of Oracle Crypto.

Oracle S/MIME

Oracle S/MIME implements the Secure/Multipurpose Internet Mail Extension (S/MIME) specifications from the Internet Engineering Task Force (IETF) for secure e-mail.

Oracle Universal Federation Framework

A unified, extensible and customizable architecture for rapid deployment of cross-domain single sign-on to federation standards in any multi-vendor environment.

The framework provides a broad array of customization features, which are described in Part III, "Oracle Universal Federation Framework".

Oracle Wallet Manager

A Java-based application that security administrators use to manage public-key security credentials on clients and servers.

See also: *Oracle Advanced Security Administrator's Guide*.

Oracle Web Services Security

Oracle Web Services Security provides a framework for authentication and authorization using existing security technologies as outlined in the OASIS specification for Web Services Security.

Oracle XML Security

Oracle XML Security implements the W3C specifications for XML Encryption and XML Signature.

OracleAS Portal

An Oracle Single Sign-On partner application that provides a mechanism for integrating files, images, applications, and Web sites. The External Applications portlet provides access to external applications.

OWM

See Oracle Wallet Manager.

partition

A unique, non-overlapping directory naming context that is stored on one directory server.

partner application

An Oracle Application Server application or non-Oracle application that delegates the authentication function to the Oracle Single Sign-On server. This type of application spares users from reauthenticating by accepting mod_osso headers.

peer-to-peer replication

Also called multimaster replication or *n*-way replication. A type of replication that enables multiple sites, acting as equals, to manage groups of replicated data. In such a replication environment, each node is both a supplier and a consumer node, and the entire directory is replicated on each node.

PKCS#1

The Public Key Cryptography Standards (PKCS) are specifications produced by RSA Laboratories. PKCS#1 provides recommendations for the implementation of public-key cryptography based on the RSA algorithm, covering the following aspects: cryptographic primitives; encryption schemes; signature schemes; ASN.1 syntax for representing keys and for identifying the schemes.

PKCS#5

The Public Key Cryptography Standards (PKCS) are specifications produced by RSA Laboratories. PKCS#5 provides recommendations for the implementation of password-based cryptography.

PKCS#7

The Public Key Cryptography Standards (PKCS) are specifications produced by RSA Laboratories. PKCS #7 describes general syntax for data that may have cryptography applied to it, such as digital signatures and digital envelopes.

PKCS#8

The Public Key Cryptography Standards (PKCS) are specifications produced by RSA Laboratories. PKCS #8 describes syntax for private key information, including a private key for some public key algorithms and a set of attributes. The standard also describes syntax for encrypted private keys.

PKCS#10

The Public Key Cryptography Standards (PKCS) are specifications produced by RSA Laboratories. PKCS #10 describes syntax for a request for certification of a public key, a name, and possibly a set of attributes.

PKCS#12

The Public Key Cryptography Standards (PKCS) are specifications produced by RSA Laboratories. PKCS #12 describes a transfer syntax for personal identity information, including private keys, certificates, miscellaneous secrets, and extensions. Systems (such as browsers or operating systems) that support this standard allow a user to import, export, and exercise a single set of personal identity information—typically in a format called a wallet.

PKI

See public key infrastructure (PKI).

plaintext

Plaintext is readable data prior to a transformation to ciphertext using encryption, or readable data that is the result of a transformation from ciphertext using decryption.

point-to-point replication

Also called fan-out replication is a type of replication in which a supplier replicates directly to a consumer. That consumer can then replicate to one or more other consumers. The replication can be either full or partial.

policy precedence

In Oracle Application Server Certificate Authority (OCA), policies are applied to incoming requests in the order that they are displayed on the main policy page. When the OCA policy processor module parses policies, those that appear toward the top of the policy list are applied to requests first. Those that appear toward the bottom of the list are applied last and take precedence over the others. Only enabled policies are applied to incoming requests.

policy.properties

A multipurpose configuration file for Oracle Single Sign-On that contains basic parameters required by the single sign-on server. Also used to configure advanced features of Oracle Single Sign-On, such as multilevel authentication.

POSIX

Portable Operating System Interface for UNIX. A set of programming interface standards governing how to write application source code so that the applications are portable between operating systems. A series of standards being developed by the Internet Engineering Task Force (IETF).

POST

An authentication method whereby login credentials are submitted within the body of the login form.

predicates

In Oracle Application Server Certificate Authority (OCA), a policy predicate is a logical expression that can be applied to a policy to limit how it is applied to incoming certificate requests or revocations. For example, the following predicate expression specifies that the policy in which it appears can have a different effect for requests or revocations from clients with DNs that include "ou=sales,o=acme,c=us":

```
Type=="client" AND DN=="ou=sales,o=acme,c=us"
```

primary node

In an Oracle Application Server Cold Failover Cluster (Identity Management), the cluster node on which the application runs at any given time.

See also: secondary node.

principal

A principal is any entity capable of using a service and capable of acquiring a federated identity.

See also: federated identity management (FIM).

private key

A private key is the secret key in a public/private key pair used in public key cryptography. An entity uses its private key to decrypt data that has been encrypted with its public key. The entity can also use its private key to create digital signatures. The security of data encrypted with the entity's public key and signatures created by the private key depends on the private key remaining secret.

private key cryptography

See symmetric cryptography.

provisioned applications

Applications in an environment where user and group information is centralized in Oracle Internet Directory. These applications are typically interested in changes to that information in Oracle Internet Directory.

provisioning

The process of providing users with access to applications and other resources that may be available in an enterprise environment.

provisioning agent

An application or process that translates Oracle-specific provisioning events to external or third-party application-specific events.

proxy server

A server between a client application, such as a Web browser, and a real server. It intercepts all requests to the real server to see if it can fulfil the requests itself. If not, it forwards the request to the real server. In Oracle Single Sign-On, proxies are used for load balancing and as an extra layer of security.

See also: load balancer.

proxy user

A kind of user typically employed in an environment with a middle tier such as a firewall. In such an environment, the end user authenticates to the middle tier. The middle tier then logs into the directory on the end user's behalf. A proxy user has the privilege to switch identities and, once it has logged into the directory, switches to the end user's identity. It then performs operations on the end user's behalf, using the authorization appropriate to that particular end user.

public key

A public key is the non-secret key in a public/private key pair used in public key cryptography. A public key allows entities to encrypt data that can only then be decrypted with the public key's owner using the corresponding private key. A public key can also be used to verify digital signatures created with the corresponding private key.

public key certificate

See certificate.

public key cryptography

Public key cryptography (also known as asymmetric cryptography) uses two keys, one public and the other private. These keys are called a key pair. The private key must be kept secret, while the public key can be transmitted to any party. The private key and the public key are mathematically related. A message that is signed by a private key can be verified by the corresponding public key. Similarly, a message encrypted by the public key can be decrypted by the private key. This method ensures privacy because only the owner of the private key can decrypt the message.

public key encryption

The process in which the sender of a message encrypts the message with the public key of the recipient. Upon delivery, the message is decrypted by the recipient using the recipient's private key.

public key infrastructure (PKI)

A public key infrastructure (PKI) is a system that manages the issuing, distribution, and authentication of public keys and private keys. A PKI typically comprises the following components:

- A Certificate Authority (CA) that is responsible for generating, issuing, publishing and revoking digital certificates.
- A Registration Authority (RA) that is responsible for verifying the information supplied in requests for certificates made to the CA.
- A directory service where a certificate or certificate revocation list (CRL) gets published by the CA and where they can be retrieved by relying third parties.
- Relying third parties that use the certificates issued by the CA and the public keys contained therein to verify digital signatures and encrypt data.

public/private key pair

A mathematically related set of two numbers where one is called the private key and the other is called the public key. Public keys are typically made widely available, while private keys are available only to their owners. Data encrypted with a public key can only be decrypted with its associated private key and vice versa. Data encrypted with a public key cannot be decrypted with the same public key.

RC2

Rivest Cipher Two (RC2) is a 64-bit block cipher developed by Ronald Rivest for RSA Security, and was designed as a replacement for Data Encryption Standard (DES).

RC4

Rivest Cipher Four (RC4) is a stream cipher developed by Ronald Rivest for RSA Security. RC4 allows variable key lengths up to 1024 bits. RC4 is most commonly used to secure data communications by encrypting traffic between Web sites that use the Secure Sockets Layer (SSL) protocol.

RDN

See relative distinguished name (RDN).

readable data

Data prior to a transformation to ciphertext via encryption or data that is the result of a transformation from ciphertext via decryption.

Registration Authority (RA)

The Registration Authority (RA) is responsible for verifying and enrolling users before a certificate is issued by a Certificate Authority (CA). The RA may assign each applicant a relative distinguished value or name for the new certificate applied. The RA does not sign or issue certificates.

relational database

A structured collection of data that stores data in tables consisting of one or more rows, each containing the same set of columns. Oracle makes it very easy to link the data in multiple tables. This is what makes Oracle a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the tables. The link is based on one or more fields common to both tables.

relative distinguished name (RDN)

The local, most granular level entry name. It has no other qualifying entry names that would serve to uniquely address the entry. In the example, `cn=Smith,o=acme,c=US`, the RDN is `cn=Smith`.

remote master site (RMS)

In a replicated environment, any site, other than the master definition site (MDS), that participates in Oracle Database Advanced Replication.

response time

The time between the submission of a request and the completion of the response.

RFC

The Internet Request For Comments (or RFC) documents are the written definitions of the protocols and policies of the Internet. The Internet Engineering Task Force (IETF) facilitates the discussion, development, and establishment of new standards. A standard is published using the RFC acronym and a reference number. For example, the official standard for e-mail is RFC 822.

root CA

In a hierarchical public key infrastructure (PKI), the root Certificate Authority (CA) is the CA whose public key serves as the most trusted datum for a security domain.

root directory specific entry (DSE)

An entry storing operational information about the directory. The information is stored in a number of attributes.

root DSE

See root directory specific entry (DSE).

root Oracle Context

In the Oracle Identity Management infrastructure, the root Oracle Context is an entry in Oracle Internet Directory containing a pointer to the default identity management realm in the infrastructure. It also contains information on how to locate an identity management realm given a simple name of the realm.

RSA

RSA is a public key cryptography algorithm named after its inventors (Rivest, Shamir, and Adelman). The RSA algorithm is the most commonly used encryption and authentication algorithm and is included as part of the Web browsers from Netscape and Microsoft, and many other products.

RSAES-OAEP

The RSA Encryption Scheme - Optimal Asymmetric Encryption Padding (RSAES-OAEP) is a public key encryption scheme combining the RSA algorithm with the OAEP method. Optimal Asymmetric Encryption Padding (OAEP) is a method for encoding messages developed by Mihir Bellare and Phil Rogaway.

S/MIME

See Secure/Multipurpose Internet Mail Extension (S/MIME).

SAML

See Security Assertions Markup Language (SAML).

SASL

See Simple Authentication and Security Layer (SASL).

scalability

The ability of a system to provide throughput in proportion to, and limited only by, available hardware resources.

schema

The collection of attributes, object classes, and their corresponding matching rules.

secondary node

In an Oracle Application Server Cold Failover Cluster (Identity Management), the cluster node to which an application is moved during a failover.

See also: primary node.

secret key

A secret key is the key used in a symmetric algorithm. Since a secret key is used for both encryption and decryption, it must be shared between parties that are transmitting ciphertext to one another but must be kept secret from all unauthorized entities.

secret key cryptography

See symmetric cryptography.

Secure Hash Algorithm (SHA)

Secure Hash Algorithm (SHA) is a hash function algorithm that produces a 160-bit message digest based upon the input. The algorithm is used in the Digital Signature Standard (DSS). With the introduction of the Advanced Encryption Standard (AES) which offers three key sizes: 128, 192 and 256 bits, there has been a need for a companion hash algorithm with a similar level of security. The newer SHA-256, SHA-284 and SHA-512 hash algorithms comply with these enhanced requirements.

Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) is a protocol designed by Netscape Communications to enable encrypted, authenticated communications across networks (such as the Internet). SSL uses the public key encryption system from RSA, which also includes the use of a digital certificate. SSL provides three elements of secure communications: confidentiality, authentication, and integrity.

SSL has evolved into Transport Layer Security (TLS). TLS and SSL are not interoperable. However, a message sent with TLS can be handled by a client that handles SSL.

Secure/Multipurpose Internet Mail Extension (S/MIME)

Secure/Multipurpose Internet Mail Extension (S/MIME) is an Internet Engineering Task Force (IETF) standard for securing MIME data through the use of digital signatures and encryption.

Security Assertions Markup Language (SAML)

Security Assertions Markup Language (SAML) is an XML-based framework for exchanging security information over the Internet. SAML enables the exchange of authentication and authorization information between various security services systems that otherwise would not be able to interoperate. The SAML 1.0 specification was adopted by OASIS in 2002.

server certificate

A certificate that attests to the identity of an organization that uses a secure Web server to serve data. A server certificate must be associated with a public/private key pair issued by a mutually trusted Certificate Authority (CA). Server certificates are required for secure communications between a browser and a Web server.

service provider

These are organizations recognized by the members of a circle of trust as the entities that provide Web-based services to users. Service providers enter into partnerships with other service providers and identity providers with the goal of providing their common users with secure single sign-on between all parties of the federation.

service time

The time between the initiation of a request and the completion of the response to the request.

session key

A secret key that is used for the duration of one message or communication session.

SGA

See System Global Area (SGA).

SHA

See Secure Hash Algorithm (SHA).

shared server

A server that is configured to allow many user processes to share very few server processes, so the number of users that can be supported is increased. With shared server configuration, many user processes connect to a dispatcher. The dispatcher directs multiple incoming network session requests to a common queue. An idle shared server process from a shared pool of server processes picks up a request from the queue. This means a small pool of server processes can server a large amount of clients. Contrast with dedicated server.

sibling

An entry that has the same parent as one or more other entries.

Signed Public Key And Challenge (SPKAC)

Signed Public Key And Challenge (SPKAC) is a proprietary protocol used by the Netscape Navigator browser to request certificates.

simple authentication

The process by which the client identifies itself to the server by means of a DN and a password which are not encrypted when sent over the network. In the simple authentication option, the server verifies that the DN and password sent by the client match the DN and password stored in the directory.

Simple Authentication and Security Layer (SASL)

A method for adding authentication support to connection-based protocols. To use this specification, a protocol includes a command for identifying and authenticating a user to a server and for optionally negotiating a security layer for subsequent protocol interactions. The command has a required argument identifying a SASL mechanism.

single key-pair wallet

A PKCS#12-format wallet that contains a single user certificate and its associated private key. The public key is imbedded in the certificate.

single sign-off

The process by which you terminate an Oracle Single Sign-On session and log out of all active partner applications simultaneously. You can do this by logging out of the application that you are working in.

single sign-on (SSO)

A process or system that enables a user to access multiple computer platforms or application systems after being authenticated only once.

single sign-on SDK

Legacy APIs to enable Oracle Single Sign-On partner applications for single sign-on. The SDK consists of PL/SQL and Java APIs and sample code that demonstrates how these APIs are implemented. This SDK is now deprecated and mod_osso is used instead.

single sign-on server

Program logic that enables users to log in securely to single sign-on applications such as expense reports, mail, and benefits.

SLAPD

Standalone LDAP daemon. An LDAP directory server service that is responsible for most functions of a directory except replication.

SOAP

Simple Object Access Protocol (SOAP) is an XML-based protocol that defines a framework for passing messages between systems over the Internet via HTTP. A SOAP message consists of three parts — an envelope that describes the message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.

specific administrative area

Administrative areas control:

- Subschema administration
- Access control administration
- Collective attribute administration

A *specific* administrative area controls one of these aspects of administration. A specific administrative area is part of an autonomous administrative area.

SPKAC

See Signed Public Key And Challenge (SPKAC).

sponsor node

In replication, the node that is used to provide initial data to a new node.

SSL

See Secure Sockets Layer (SSL).

stream cipher

Stream ciphers are a type of symmetric algorithm. A stream cipher encrypts in small units, often a bit or a byte at a time, and implements some form of feedback mechanism so that the key is constantly changing. RC4 is an example of a stream cipher.

subACLSubentry

A specific type of subentry that contains access control list (ACL) information.

subclass

An object class derived from another object class. The object class from which it is derived is called its superclass.

subentry

A type of entry containing information applicable to a group of entries in a subtree. The information can be of these types:

- Access control policy points
- Schema rules
- Collective attributes

Subentries are located immediately below the root of an administrative area.

subordinate CA

In a hierarchical public key infrastructure (PKI), the subordinate Certificate Authority (CA) is a CA whose certificate signature key is certified by another CA, and whose activities are constrained by that other CA.

subordinate reference

A knowledge reference pointing downward in the directory information tree (DIT) to a naming context that starts immediately below an entry

subschema DN

The list of directory information tree (DIT) areas having independent schema definitions.

subSchemaSubentry

A specific type of subentry containing schema information.

subtree

A section of a directory hierarchy, which is also called a directory information tree (DIT). The subtree typically starts at a particular directory node and includes all subdirectories and objects below that node in the directory hierarchy.

subtype

An attribute with one or more options, in contrast to that same attribute without the options. For example, a `commonName` (`cn`) attribute with American English as an option is a subtype of the `commonName` (`cn`) attribute without that option. Conversely, the `commonName` (`cn`) attribute without an option is the supertype of the same attribute with an option.

success URL

When using Oracle Single Sign-On, the URL to the routine responsible for establishing the session and session cookies for an application.

super user

A special directory administrator who typically has full access to directory information.

superclass

The object class from which another object class is derived. For example, the object class `person` is the superclass of the object class `organizationalPerson`. The latter, namely, `organizationalPerson`, is a subclass of `person` and inherits the attributes contained in `person`.

superior reference

A knowledge reference pointing upward to a directory system agent (DSA) that holds a naming context higher in the directory information tree (DIT) than all the naming contexts held by the referencing DSA.

supertype

An attribute without options, in contrast to the same attribute with one or more options. For example, the `commonName` (`cn`) attribute without an option is the supertype of the same attribute with an option. Conversely, a `commonName` (`cn`) attribute with American English as an option is a subtype of the `commonName` (`cn`) attribute without that option.

supplier

In replication, the server that holds the master copy of the naming context. It supplies updates from the master copy to the consumer server.

symmetric algorithm

A symmetric algorithm is a cryptographic algorithm that uses the same key for encryption and decryption. There are essentially two types of symmetric (or secret key) algorithms — stream ciphers and block ciphers.

symmetric cryptography

Symmetric cryptography (or shared secret cryptography) systems use the same key to encipher and decipher data. The problem with symmetric cryptography is ensuring a secure method by which the sender and recipient can agree on the secret key. If a third party were to intercept the secret key in transit, they could then use it to decipher anything it was used to encipher. Symmetric cryptography is usually faster than asymmetric cryptography, and is often used when large quantities of data need to be exchanged. DES, RC2, and RC4 are examples of symmetric cryptography algorithms.

symmetric key

See secret key.

System Global Area (SGA)

A group of shared memory structures that contain data and control information for one Oracle database instance. If multiple users are concurrently connected to the same instance, the data in the instance SGA is shared among the users. Consequently, the SGA is sometimes referred to as the "shared global area." The combination of the background processes and memory buffers is called an Oracle instance.

system operational attribute

An attribute holding information that pertains to the operation of the directory itself. Some operational information is specified by the directory to control the server, for example, the time stamp for an entry. Other operational information, such as access information, is defined by administrators and is used by the directory program in its processing.

think time

The time the user is not engaged in actual use of the processor.

third-party access management system

Non-Oracle single sign-on system that can be modified to use Oracle Single Sign-On to gain access to Oracle Application Server applications.

throughput

The number of requests processed byOracle Internet Directory for each unit of time. This is typically represented as "operations per second."

Time Stamp Protocol (TSP)

Time Stamp Protocol (TSP), as specified in RFC 3161, defines the participating entities, the message formats, and the transport protocol involved in time stamping a digital message. In a TSP system, a trusted third-party Time Stamp Authority (TSA) issues time stamps for messages.

TLS

See Transport Layer Security (TLS).

Transport Layer Security (TLS)

A protocol providing communications privacy over the Internet. The protocol enables client/server applications to communicate in a way that prevents eavesdropping, tampering, or message forgery.

Triple Data Encryption Standard (3DES)

Triple Data Encryption Standard (3DES) is based on the Data Encryption Standard (DES) algorithm developed by IBM in 1974, and was adopted as a national standard in 1977. 3DES uses three 64-bit long keys (overall key length is 192 bits, although actual key length is 56 bits). Data is encrypted with the first key, decrypted with the second key, and finally encrypted again with the third key. This makes 3DES three times slower than standard DES but also three times more secure.

trusted certificate

A third party identity that is qualified with a level of trust. The trust is used when an identity is being validated as the entity it claims to be. Typically, trusted certificates come from a Certificate Authority (CA) you trust to issue user certificates.

trustpoint

See trusted certificate.

TSP

See Time Stamp Protocol (TSP).

Unicode

A type of universal character set, a collection of 64K characters encoded in a 16-bit space. It encodes nearly every character in just about every existing character set standard, covering most written scripts used in the world. It is owned and defined by Unicode Inc. Unicode is canonical encoding which means its value can be passed around in different locales. But it does not guarantee a round-trip conversion between it and every Oracle character set without information loss.

UNIX Crypt

The UNIX encryption algorithm.

URI

Uniform Resource Identifier (URI). A way to identify any point of content on the Web, whether it be a page of text, a video or sound clip, a still or animated image, or a program. The most common form of URI is the Web page address, which is a particular form or subset of URI called a URL.

URL

Uniform Resource Locator (URL). The address of a file accessible on the Internet. The file can be a text file, HTML page, image file, a program, or any other file supported by HTTP. The URL contains the name of the protocol required to access the resource, a domain name that identifies a specific computer on the Internet, and a hierarchical description of the file location on the computer.

URLC token

The Oracle Single Sign-On code that passes authenticated user information to the partner application. The partner application uses this information to construct the session cookie.

user name mapping module

A Oracle Single Sign-On Java module that maps a user certificate to the user's nickname. The nickname is then passed to an authentication module, which uses this nickname to retrieve the user's certificate from the directory.

user search base

In the Oracle Internet Directory default directory information tree (DIT), the node in the identity management realm under which all the users are placed.

UTC (Coordinated Universal Time)

The standard time common to every place in the world. Formerly and still widely called Greenwich Mean Time (GMT) and also World Time, UTC nominally reflects the mean solar time along the Earth's prime meridian. UTC is indicated by a z at the end of the value, for example, 200011281010z.

UTF-8

A variable-width 8-bit encoding of Unicode that uses sequences of 1, 2, 3, or 4 bytes for each character. Characters from 0-127 (the 7-bit ASCII characters) are encoded with one byte, characters from 128-2047 require two bytes, characters from 2048-65535 require three bytes, and characters beyond 65535 require four bytes. The Oracle character set name for this is AL32UTF8 (for the Unicode 3.1 standard).

UTF-16

16-bit encoding of Unicode.The Latin-1 characters are the first 256 code points in this standard.

verification

Verification is the process of ensuring that a given digital signature is valid, given the public key that corresponds to the private key purported to create the signature and the data block to which the signature purportedly applies.

virtual host

A single physical Web server machine that is hosting one or more Web sites or domains, or a server that is acting as a proxy to other machines (accepts incoming requests and reroutes them to the appropriate server).

In the case of Oracle Single Sign-On, virtual hosts are used for load balancing between two or more Oracle Single Sign-On servers. They also provide an extra layer of security.

virtual host name

In an Oracle Application Server Cold Failover Cluster (Identity Management), the host name corresponding to a particular virtual IP address.

virtual IP address

In an Oracle Application Server Cold Failover Cluster (Identity Management), each physical node has its own physical IP address and physical host name. To present a single system image to the outside world, the cluster uses a dynamic IP address that can be moved to any physical node in the cluster. This is called the virtual IP address.

wait time

The time between the submission of the request and initiation of the response.

wallet

An abstraction used to store and manage security credentials for an individual entity. It implements the storage and retrieval of credentials for use with various cryptographic services. A wallet resource locator (WRL) provides all the necessary information to locate the wallet.

Wallet Manager

See Oracle Wallet Manager.

Web service

A Web service is application or business logic that is accessible using standard Internet protocols, such as HTTP, XML, and SOAP. Web Services combine the best aspects of component-based development and the World Wide Web. Like components, Web

Services represent black-box functionality that can be used and reused without regard to how the service is implemented.

Web Services Description Language (WSDL)

Web Services Description Language (WSDL) is the standard format for describing a Web service using XML. A WSDL definition describes how to access a Web service and what operations it will perform.

WSDL

See Web Services Description Language (WSDL).

WS-Federation

Web Services Federation Language (WS-Federation) is a specification developed by Microsoft, IBM, VeriSign, and RSA Security. It defines mechanisms to allow federation between entities using different or like mechanisms by allowing and brokering trust of identities, attributes, and authentication between participating Web services.

See also: Liberty Alliance.

X.500

X.500 is a standard from the International Telecommunication Union (ITU) that defines how global directories should be structured. X.500 directories are hierarchical with different levels for each category of information, such as country, state, and city.

X.509

X.509 is the most widely used standard for defining digital certificates. A standard from the International Telecommunication Union (ITU), for hierarchical directories with authentication services, used in many public key infrastructure (PKI) implementations.

XML

Extensible Markup Language (XML) is a specification developed by the World Wide Web Consortium (W3C). XML is a pared-down version of Standard Generalized Mark-Up Language (SGML), designed especially for Web documents. XML is a metalanguage (a way to define tag sets) that allows developers to define their own customized markup language for many classes of documents.

XML canonicalization (C14N)

This is a process by which two logically equivalent XML documents can be resolved to the same physical representation. This has significance for digital signatures because a signature can only verify against the same physical representation of the data against which it was originally computed. For more information, see the W3C's XML Canonicalization specification.

Index