Oracle® Fusion Middleware

Security and Administrator's Guide for Web Services

11*g* Release 1 (11.1.1.9)

E78149-02

March 2018

This document describes how to administer and secure Web services.

Oracle Fusion Middleware Security and Administrator's Guide for Web Services, 11*g* Release 1 (11.1.1.9)

E78149-02

Primary Author: Oracle Corporation

Contents

3 Understanding Oracle WSM Policy Framework

4 Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware

Part II Basic Administration

5 Deploying Web Services Applications

6 Administering Web Services

7 Managing Web Service Policies

8 Attaching Policies to Web Services

9 Creating and Managing Policy Sets

10 Setting Up Your Environment for Policies

11 Configuring Policies

12 Testing Web Services

13 Monitoring the Performance of Web Services

Part III Advanced Administration

14 Advanced Administration

15 Managing Application Migration Between Environments

16 Diagnosing Problems

17 Maintaining the Oracle WSM Repository

Part IV WebLogic Web Service Administration

18 Securing and Administering WebLogic Web Services

Part V Reference

A Web Service Security Standards

B Predefined Policies

C Predefined Assertion Templates

D Schema Reference for Predefined Assertions

E Schema Reference for Policy Sets

Preface

This section describes the intended audience, how to use this guide, and provides information about documentation accessibility.

About this Guide

This guide describes the tasks required to secure and administer Web services, providing details describing how to:

- Deploy, configure, test, and monitor Web services.
- Enable, publish, and register Web services.
- Attach policies for security, messaging, addressing, and management of Web services, and analyzing policy usage.
- Create new policies and assertion templates, and manage and configure existing policies.
- Manage policy lifecycle to transition from a test to production environment.
- Manage your file-based and database stores in your development and production environments, respectively.
- Diagnose problems.

Audience

This guide is intended for:

- System and security administrators who administer Web services and manage security
- Application developers who are developing Web services and testing the security prior to deployment of the Web services
- Security architects who create security policies

How to Use This Guide

It is recommended that you review *Oracle Fusion Middleware Introducing Web Services* document to gain a better understanding of the two Web service stacks supported in Oracle Fusion Middleware 11*g*.

The document is organized as follows:

- Part I, "Introduction" introduces you to the concepts and tasks required to secure and administer Web services, and describes a set of common use cases.

Chapter 4, "Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware" discusses how the features of Oracle WSM have been rearchitected in Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9). If you are an existing Oracle Web Services Manager 10*g* (Oracle WSM) customer, it is recommended that you review this chapter.

- Part II, "Basic Administration" describes the basic administration tasks that you can perform, such as deploying and configuring Web services; managing and attaching, and configuring policies; testing and monitoring Web services, and more.
- Part III, "Advanced Administration" describes the advanced administration tasks such as publishing and auditing Web services; migrating from a file-based store; managing policy lifecycle, diagnosing problems, and more.
- Part IV, "WebLogic Web Service Administration" describes how to secure and administer WebLogic (Java EE) Web services.
- Part V, "Reference" provides reference information describing Web service security standards; predefined policy and assertion templates; and assertion schemas.

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 11*g* Release 1 (11.1.1.9) documentation set:

- *Introducing Web Services*
- *Developer's Guide for Oracle Infrastructure Web Services*
- *Introducing WebLogic Web Services for Oracle WebLogic Server*
- *Extensibility Guide for Oracle Web Services Manager*
- *Interoperability Guide for Oracle Web Services Manager*
- *Getting Started With JAX-WS Web Services for Oracle WebLogic Server*
- *Programming Advanced Features of JAX-WS Web Services for Oracle WebLogic Server*
- *Getting Started With JAX-RPC Web Services for Oracle WebLogic Server*
- *Programming Advanced Features of JAX-RPC Web Services for Oracle WebLogic Server*
- *Securing WebLogic Web Services for Oracle WebLogic Server*
- *WebLogic Web Services Reference for Oracle WebLogic Server*
- *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*

- *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*
- "Developing with Web Services" in the Oracle JDeveloper online help

See also the *Oracle Web Services Manager* Technology page at: `http://www.oracle.com/technology/products/webservices_manager/index.html`.

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

11g Release 1 includes a complete redesign of Oracle Web Services Manager 10*g* and Web services security management. For more details about what has changed in Release 11*g*, see Chapter 4, "Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware."

The following topics provide a summary of the features and enhancements in each of the 11*g* Release 1 releases:

- 11g Release 1 (11.1.1.9)
- 11g Release 1 (11.1.1.7)
- 11g Release 1 (11.1.1.6)
- 11g Release 1 (11.1.1.5)
- 11g Release 1 (11.1.1.4)
- 11g Release 1 (11.1.1.3)
- 11g Release 1 (11.1.1.2)
- 11g Release 1 (11.1.1)

11*g* Release 1 (11.1.1.9)

11g Release 1 (11.1.1.9) includes the following new features and enhancements:

- New Features
- New WLST Commands
- New Predefined Polices

New Features

The following new features and enhancements have been added to the current release of Oracle Web Services Manager:

- Upgrade Oracle WSM Configuration to Release 11.1.1.9
- Ability to Configure an Application-level Credential Map
- OAuth 2.0 Support for REST and SOAP Services and Clients
- RESTful Client Support and Security
- REST APIs for Managing Credentials and Keystores
- Token Attribute Rules

- WLST Commands for Managing Distinguished Name (DN) Lists
- WLST Commands for Managing a Java Keystore (JKS)
- Client Policies that Enable Identity Switching for Web Service Clients

Upgrade Oracle WSM Configuration to Release 11.1.1.9

The `upgradeWSM` WLST command upgrades the OWSM configuration in a WebLogic Server 11*g* domain from a previous release (11.1.1.1.0–11.1.1.6.0) to Release 11.1.1.9.0. This command is documented in the following locations:

- Upgrading Oracle WSM Configurations" in the *Oracle Fusion Middleware Patching Guide*
- "Upgrade OWSM Configuration Command" in the *WebLogic Scripting Tool Command Reference*

Ability to Configure an Application-level Credential Map

An application-level credential map name can be set in certain predefined policies using the csf.map configuration property, which can be used to override the domain-level credential map on a per-attachment basis. The csf.map configuration override is available in all policies and assertion templates that have either a csf-key or keystore-related csf keys.

For more information, see "Creating an Application-level Credential Map" on page 10-24.

OAuth 2.0 Support for REST and SOAP Services and Clients

Oracle WSM allows web service clients to interact with the Mobile and Social OAuth 2.0 server implementation for both SOAP and REST web services, for "2-legged" authorization.

For more information, see "Using OAuth2 with Oracle WSM" on page 10-74.

RESTful Client Support and Security

To secure RESTful Web service clients, attach Oracle WSM policies globally using WLST.

For more information, see "Attaching Policies to RESTful Web Service Clients Using WLST" on page 8-18.

REST APIs for Managing Credentials and Keystores

The credential and keystore management REST API provides endpoints for creating and configuring credential stores, keystores, and trust stores for your domain or Web services. For more information, see *REST API for Managing Credentials and Keystores with Oracle Web Services Manager*.

Token Attribute Rules

In addition to the token attribute rules introduced in Release 11.1.1.6.0 to apply additional security constraints for the trusted STS (Secure Token Service) server and for the trusted SAML client, Oracle WSM allows you to define token attribute rules for trusted JWT clients. Also, a new token attribute mapping rule so the name ID attribute can map a local user attribute for the subject name ID to the local user attributes to authenticate a trusted user.

Token attribute rules can be applied through the Fusion Middleware Control or by using WLST commands. For more information, see "Configuring Token Attribute

Rules for Trusted Issuers" on page 14-28.

WLST Commands for Managing Distinguished Name (DN) Lists

Oracle WSM adds the ability to manage DN lists with WLST commands. There are three new WLST commands to revoke trust by removing trusted issuer configurations (DNs and token attribute rules), exporting trust configurations to an XML, and importing trust configurations from an XML file. These commands include `revokeWSMTokenIssuerTrust`, `exportWSMTokenIssuerTrustMetadata`, and `importWSMTokenIssuerTrustMetadata`.

For more information, see "Defining Trusted Issuers and Managing DN Lists Using WLST" on page 14-25.

WLST Commands for Managing a Java Keystore (JKS)

The following WLST commands were added to manage and view JKS credentials and certificates:

- `deleteWSMKeyStoreEntry`
- `deleteWSMKeyStoreEntries`
- `exportWSMCertificate`
- `importWSMCertificate`
- `listWSMKeystoreAliases`
- `displayWSMCertificate`

For more information, see "Managing Java Keystore Certificates" on page 10-16.

Client Policies that Enable Identity Switching for Web Service Clients

The current release includes the following new predefined policies. For more information, see Appendix B, "Predefined Policies."

To support identity switching for web service clients, the following predefined SAML and JWT client policies are provided:

- oracle/wss_saml_token_bearer_identity_switch_client_policy – This policy can be attached to any SOAP client endpoint.
- oracle/http_jwt_token_identity_switch_client_policy – This policy can be enforced on any HTTP-based, SOAP, or REST client endpoint.

For more information, see "Configuring Web Service Clients for Identity Switching" on page 10-78.

New WLST Commands

The current release adds these Web Services WLST commands. For more information on these commands, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

The following commands are associated with managing DN lists. For more information, see "Defining Trusted Issuers and Managing DN Lists Using WLST" on page 14-25.

- `revokeWSMTokenIssuerTrust`—revokes trust by removing all trusted issuers and associated configurations (DNs and token attribute rules). See "Revoking Trust from Trusted Issuers Using WLST" on page 14-28.

- `exportWSMTokenIssuerTrustMetadata`—exports all the trust configurations (issuer, DNs, and token attribute rules) for all trusted issuers to an XML file. See "Exporting Trust Metadata Using WLST" on page 14-27.
- `importWSMTokenIssuerTrustMetadata`—imports the trust configurations (issuers, DNs, and token attribute rules) for all trusted issuers from an XML file. See "Importing Trust Metadata Using WLST" on page 14-27.
- `setWSMTokenIssuerTrustAttributeMapping`—maps the name ID attribute of a local user attribute for the subject name ID to the local user attributes to authenticate a trusted user. See "Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control" on page 14-29.

The following commands were added to manage and view JKS credentials and certificates. For more information, see "Managing Java Keystore Certificates" on page 10-16.

- `deleteWSMKeyStoreEntry`—deletes a single certificate from the keystore.
- `deleteWSMKeyStoreEntries`—deletes all certificates from the keystore.
- `exportWSMCertificate`—exports a trusted certificate or a certificate chain associated with a private key, indicated by a specified alias, to a specified location.
- `importWSMCertificate`—imports a trusted certificate or a certificate chain associated with a private key indicated by a specified alias.
- `listWSMKeystoreAliases`—lists all the aliases in the keystore.
- `displayWSMCertificate`—displays the string representing the contents of a user's certificate if the alias specifies a `KeyStore.TrustedCertificateEntry`. Displays the certificates in the chain if the alias points to a certificate chain specified by the `KeyStore.PrivateKeyEntry`.

The following command was added to upgrade a OWSM configuration in a WebLogic Server 11*g* domain from a previous release (11.1.1.1.0–11.1.1.6.0) to Release 11.1.1.9.0. For more information, see Upgrading Oracle WSM Configurations" in the *Oracle Fusion Middleware Patching Guide*.

- `upgradeWSM`—deletes a single certificate from the keystore.

New Predefined Polices

The current release includes the following new predefined policies. For more information, see Appendix B, "Predefined Policies."

To support identity switching for web service clients, the following predefined SAML and JWT client policies are provided:

- oracle/wss_saml_token_bearer_identity_switch_client_policy – This policy can be attached to any SOAP client endpoint.
- oracle/http_jwt_token_identity_switch_client_policy – This policy can be enforced on any HTTP-based, SOAP, or REST client endpoint.

For more information, see "Configuring Web Service Clients for Identity Switching" on page 10-78.

11*g* Release 1 (11.1.1.7)

11g Release 1 (11.1.1.7) includes the following new features and enhancements:

- New Features

- New WLST Commands
- New Predefined Policies
- New Predefined Assertion Templates

New Features

The following new features and enhancements have been added to the current release of Oracle Web Services Manager:

- Support for JSON Web Tokens (JWT) for Identity Propagation
- New Section for Troubleshooting WS-Trust Configurations
- Check the Status of Oracle WSM Components
- Token Attribute Rules
- WLST Commands for Managing Distinguished Name (DN) Lists
- Policy Accessor Properties for Tuning the Repository Connection
- ID Context Propagation
- Automatic Oracle WSM Repository Upgrade After Patch Set Installation
- WLST Commands to Attach Policies to Java EE Web Services
- Keystore Service (KSS) Enhancements
- Servlet Application Security
- Interoperability of Oracle WSM with .NET AND ADFS 2.0 STS
- Ability to Sign and Encrypt SOAP Parts and Elements in Fault Messages

Support for JSON Web Tokens (JWT) for Identity Propagation

Oracle WSM now includes support for JSON Web Token (JWT) as a means of representing claims to be transferred between two parties. JWT is a compact token format intended for space-constrained environments such as HTTP Authorization headers.

References to the JWT token have been added throughout the document. Additional information is provided in the following sections:

- "Using JSON Web Token (JWT) with Oracle WSM" on page 10-69
- "Attaching Policies to Servlet Applications" on page 8-18
- "Propagating Identity Context with Oracle WSM" on page 10-81
- "Configuring a Policy With an OR Group" on page 11-34
- "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23, specifically "Defining Trusted Issuers and Managing DN Lists Using WLST" on page 14-25

The following new policies were added to support JWT:

- "oracle/http_jwt_token_client_policy"—includes a JWT token in the HTTP header.
- "oracle/http_jwt_token_service_policy"—authenticates users using the username provided in the JWT token in the HTTP header.
- "oracle/http_jwt_token_over_ssl_client_policy"—includes a JWT token in the HTTP header and verifies that the transport protocol is HTTPS.

- "oracle/http_jwt_token_over_ssl_client_policy"—authenticates users using the username provided in the JWT token in the HTTP header and verifies that the transport protocol is HTTPS.

The following new assertion templates were added to support JWT:

- "oracle/http_jwt_token_client_template"—includes a JWT token in the HTTP header.
- "oracle/http_jwt_token_service_template"—authenticates users using the username provided in the JWT token in the HTTP header.
- "oracle/http_jwt_token_over_ssl_client_template"—includes a JWT token in the HTTP header and verifies that the transport protocol is HTTPS.
- "oracle/http_jwt_token_over_ssl_service_template"—authenticates users using the username provided in the JWT token in the HTTP header and verifies that the transport protocol is HTTPS.

The following WLST commands were updated to include support for JWT tokens:

- `displayWSMTokenIssuerTrust`
- `setWSMTokenIssuerTrust`
- `deleteWSMTokenIssuerTrust`

Details for using these commands are provided in "Defining Trusted Issuers and Managing DN Lists Using WLST".

New Section for Troubleshooting WS-Trust Configurations

A new section has been added to assist in troubleshooting WS-Trust configurations. For more information, see "Diagnosing Common Oracle WSM Exceptions for WS-Trust Use Cases" on page 16-18.

Check the Status of Oracle WSM Components

Oracle WSM has added the `checkWSMStatus` WLST command which allows you to check the configuration of your domain. The `checkWSMStatus` command returns the status of the policy manager (`wsm-pm`), the agent (`agent`), and the credential store and keystore configuration (`credstore`). The status of the components can be checked together or individually.

For more information, see "Diagnosing Problems With a Domain Configuration using WLST" on page 16-16.

Token Attribute Rules

There are increasing requirements to control which users and user attributes are accepted and processed for a particular trusted user. Oracle WSM allows you to define token attribute rules to apply additional security constraints for the trusted STS (Secure Token Service) server and for the trusted SAML client. Token attribute rules can be applied through the Fusion Middleware Control or by using WLST commands.

For more information, see "Configuring Token Attribute Rules for Trusted Issuers" on page 14-28.

WLST Commands for Managing Distinguished Name (DN) Lists

Oracle WSM adds the ability to manage DN lists with WLST commands. There are new WLST commands to configure an issuer and its DN list, display the issuers and DN lists, and delete an issuer and its DN list. These commands include `deleteWSMTokenIssuerTrust`, `deleteWSMTokenIssuerTrustAttributeRule`, `setWSMTokenIssuerTrust`, `setWSMTokenIssuerTrust`, and `setWSMTokenIssuerTrust`.

For more information, see "Defining Trusted Issuers and Managing DN Lists Using WLST" on page 14-25.

Policy Accessor Properties for Tuning the Repository Connection

New properties have been added to the Policy Accessor to enable you to configure the connection between the Agent and the Policy Manager. Some of the things the properties allow you to configure include how often the runtime attempts to reconnect to the Policy Manager, the number of times the Agent will attempt to communicate with the Policy Manager (which in turn accesses the Repository) and the time interval between retries, and how often the Agent attempts to contact the Policy Manager to refresh documents it has already cached.

For more information, see"Tuning WSM Repository Connections" on page 14-19.

ID Context Propagation

Identity Context allows applications in a system to have visibility into a shared identity context to manage identity-related risks in their security policies. Oracle WSM propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes.

For more information, see "Propagating Identity Context with Oracle WSM" on page 10-81.

Automatic Oracle WSM Repository Upgrade After Patch Set Installation

After you install a Fusion Middleware patch set, the repository is automatically upgraded, as part of the server startup process, with the latest predefined policies and assertion templates. You no longer need to execute the `upgradeWSMPolicyRepository` command.

For more information, see "Upgrading the Oracle WSM Policies in the Repository" on page 17-7.

WLST Commands to Attach Policies to Java EE Web Services

In Oracle WSM you can now perform policy attachment and detachment operations on Java EE Web services and clients using WLST commands. See the following sections:

- "Attaching a Policy to a Web Service Using WLST" on page 8-6 for information on the `attachWebServicePolicy`, `attachWebServicePolicies`, `detachWebServicePolicy`, and `detachWebServicePolicies` commands.
- "Attaching Policies to Web Service Clients Using WLST" on page 8-15 for information on the `attachWebServiceClientPolicy`, `attachWebServiceClientPolicies`, `detachWebServiceClientPolicy`, and `detachWebServiceClientPolicies` commands.

Keystore Service (KSS) Enhancements

As described in "Managing Keys and Certificates with the Keystore Service" in *Oracle Fusion Middleware Application Security Guide*, the Oracle Platform Security Services (OPSS) Keystore Service provides an alternate mechanism to manage keys and certificates for message security.

For more information on how to configure the OPSS Keystore Service for message protection, see Chapter 10, "Setting Up Your Environment for Policies"of *Security and Administrator's Guide for Web Services*.

Servlet Application Security

To secure servlet applications, such as ADF business components exposed as RESTful servlets, you can attach a subset of Oracle WSM predefined security policies.

For more information, see "Attaching Policies to Servlet Applications" on page 8-18.

Interoperability of Oracle WSM with .NET AND ADFS 2.0 STS

The "Interoperability with Microsoft WCF/.NET 3.5 Security Environments" chapter of *Interoperability Guide for Oracle Web Services Manager* now provides instructions for securing WCF/.NET 3.5 Client with Microsoft Active Directory Federation Services (ADFS) 2.0.

Ability to Sign and Encrypt SOAP Parts and Elements in Fault Messages

Oracle WSM now supports signing and encrypting body parts and header elements in fault messages for message protection policies. By default fault protection is disabled. You can configure this setting in the Message Security section of the message protection policies. For more information, see "Message Signing and Encryption Settings for Request, Response, and Fault Messages" on page C-192.

New WLST Commands

The current release adds these Web Services WLST commands. For more information on these commands, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

The following commands are associated with managing DN lists. For more information, see "Defining Trusted Issuers and Managing DN Lists Using WLST" on page 14-25.

- `deleteWSMTokenIssuerTrust`—deletes the trusted issuer, including the DN list in it. See "Deleting an Issuer and its DN List using WLST" on page 14-27.
- `deleteWSMTokenIssuerTrustAttributeRule`—deletes a token attribute rule associated with a DN list. See "Deleting a Token Attribute Rule Using WLST" on page 14-32.
- `displayWSMTokenIssuerTrust`—displays the names of the DN lists associated with a specified issuer. See "Displaying Issuers and DN Lists using WLST" on page 14-26.
- `setWSMTokenIssuerTrust`—specifies a trusted SAML issuer with a DN list. See "Configuring an Issuer and its DN List Using WLST" on page 14-25.
- `setWSMTokenIssuerTrustAttributeFilter`—specifies the DN of a token signing certificate and a list of trusted users. See "Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control" on page 14-29.

The following command checks the status of Oracle WSM Components. For more information, see "Diagnosing Problems With a Domain Configuration using WLST" on page 16-16.

- `checkWSMStatus`—checks the status of the WSM components which are required for proper functioning of the product. See "Diagnosing Problems With a Domain Configuration using WLST" on page 16-16.

New Predefined Policies

The current release includes the following new predefined policies. For more information, see Appendix B, "Predefined Policies."

To support servlet application security, the following predefined policies are provided:

- oracle/http_oam_token_service_policy—verifies that the OAM agent has authenticated the user and has established an identity.
- oracle/http_basic_auth_over_ssl_client_policy—includes credentials in the HTTP header for outbound client requests.
- oracle/http_basic_auth_over_ssl_service_policy —uses the credentials in the HTTP header and authenticates users against the Oracle Platform Security Services identity store.
- oracle/http_saml20_token_bearer_client_policy—includes SAML 2.0 tokens in the HTTP header.
- oracle/http_saml20_token_bearer_service_policy—includes a SAML 2.0 token with confirmation method *Bearer* in the HTTP header.
- oracle/http_saml20_token_bearer_over_ssl_client_policy—includes SAML 2.0 tokens in the HTTP header.
- oracle/http_saml20_token_bearer_over_ssl_service_policy—includes a SAML 2.0 token with confirmation method *Bearer* in the HTTP header.
- oracle/multi_token_rest_service_policy—enforces an authentication policy based on the token sent by the client.
- oracle/multi_token_over_ssl_rest_service_policy—enforces an authentication policy based on the token sent by the client.

To support SAML token bearer authentication, the following predefined policies are provided:

- oracle/wss_saml_token_bearer_client_policy—includes SAML tokens in outbound SOAP request messages.
- oracle/wss_saml_bearer_or_username_token_service_policy—enforces one an authentication policy, based on whether the client uses a SAML or username token.

New Predefined Assertion Templates

The current release includes the following new predefined assertion templates. For more information, see Appendix C, "Predefined Assertion Templates."

To support servlet application security, the following predefined assertion templates are provided:

- oracle/http_oam_token_service_template —verifies that OAM agent has authenticated the user and has established an identity.
- oracle/http_saml20_token_bearer_client_template—includes SAML 2,0 tokens in outbound SOAP request messages.
- oracle/http_saml20_token_bearer_service_template—authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.
- oracle/http_spnego_token_client_template—provides authentication using a Kerberos token and the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) protocol.

- oracle/http_spnego_token_service_template—provides authentication using a Kerberos token and the SPNEGO protocol.

To support SAML token bearer authentication, the following predefined assertion templates are provided:

- oracle/wss_saml_token_bearer_client_template—includes SAML tokens in outbound SOAP request messages.
- oracle/wss_saml_token_bearer_service_template—authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

11*g* Release 1 (11.1.1.6)

11g Release 1 (11.1.1.6) includes the following new features and enhancements:

Global Policy Attachment Enhancements

The global policy attachment feature has been enhanced as follows:

- Support for attaching policies globally at the partition, service or reference, and port and component levels for clients and services. For more information, see "Subject Types and Scope of Resources" on page 9-3.
- Support for a new WLST command (`deleteAllPolicySets`) that allows a user to delete all policy set documents in the repository. For more information, see "Deleting Policy Sets" on page 9-32.
- Support for configuration overrides for global policy attachments. For more information, see "Overriding Configuration Properties for Globally Attached Policies" on page 9-22.
- Ability to specify the priority of a policy attachment which allows an administrator to indicate a preference over which policy attachment is used. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
- Improved visibility into the endpoint configuration using Fusion Middleware Control, including the ability to see the secure status of the endpoints, any configuration overrides, and if the endpoints have a valid configuration. For more information, see "Determining the Secure Status of an Endpoint" on page 9-36.

Run-Time Constraints

Oracle WSM provides the ability to specify a run-time constraint that determines the context in which the policy set is relevant, for example external clients outside a firewall versus internal clients. For more information, see "Specifying Run-time Constraints in Policy Sets" on page 9-26.

Oracle SPARC Server T-Series Cryptographic Acceleration Support

Ability to configure Oracle WSM to take advantage of Oracle SPARC Server Cryptographic Acceleration. For more information, see "Configuring Oracle WSM for Oracle SPARC T4 Cryptographic Acceleration" on page 10-51.

Enhanced Support for WebLogic Java EE Clients in Fusion Middleware Control

Ability to use Fusion Middleware Control to view and monitor Java EE clients and attach Oracle WSM policies.

- A new tab, Java EE Web Service Clients, has been added to the Web Services (Java EE) Home page for viewing information about Java EE clients. For more information, see "Viewing Java EE Web Service Clients" on page 6-10.
- Ability to attach Oracle WSM policies to Java EE clients. For more information, see "Attaching Policies to Java EE Web Service Clients" on page 8-13.
- Ability to view Web Service statistics for the run-time client instances in a Java EE application. For more information, see "Viewing Web Service Statistics for Java EE Web Service Clients" on page 13-5.

Test Web Service Enhancements

Enhanced ability to test Web service security using Oracle WSM policies. For more information, see Chapter 12, "Testing Web Services."

Derived Keys and Encrypt Signature Controls Enabled in Fusion Middleware Control

Oracle WSM supports the Derived Key setting in wss11 message protection policies and the Encrypt Signature setting in wss10 and wss11 message protection policies. You can now enable these features using Fusion Middleware Control in the Message Security settings in message protection policies. For more information about these settings, refer to the message protection assertion templates described in Appendix C, "Predefined Assertion Templates."

No Server Restart Required for JKS Keystore Changes

You no longer need to restart the server when you make changes to the JKS keystore. For more information about the JKS keystore, see "Generating Private Keys and Creating the Java Keystore" on page 10-9.

Support for Anonymous User with SAML Policies

Oracle WSM supports propagating the anonymous user with SAML policies. For more information, see "Using Anonymous Users with SAML Policies" on page 10-69.

Database Support

Oracle WSM is certified with MySQL and Oracle Edition Based Redefinitions (EBR).

Versioned Web Services

Oracle WSM supports multiple versions (namespaces) of a Web service. Service names in WLST input and output, and Fusion Middleware Control, now require the use of the namespace with the service name, for example `{http://mynamespace/}myService`. For more information, see the following topics:

- "Specifying a Service Name" in "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*
- `listWebServices` in "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*
- "Determining the Namespace for a Web Service" on page 9-21

SAML Issuer Changes

You no longer need to define SAML issuers in the SAML login module. In this release, if you define a SAML issuer using the Platform Policy Configuration page, any issuers added in the SAML login module are ignored. Also, when SAML issuers are added using the platform policy configuration, you do not need to restart the server. For more information, see "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23.

Additional OR Groups Added to wss11_saml_or_username_token_with_message_protection_service_policy

The oracle/wss11_saml_or_username_token_with_message_protection_service_policy now includes five assertions:

- wss11_saml_token_with_message_protection
- wss11_username_token_with_message_protection
- wss_saml_token_bearer_over_ssl
- wss_username_token_over_ssl
- wss_http_token_over_ssl

For more information, see "Configuring a Policy With an OR Group" on page 11-34.

11*g* Release 1 (11.1.1.5)

11*g* Release 1 (11.1.1.5) includes the following updates and enhancements:

- Added two new attributes to the asynchronous Web service queue annotations, `@AsyncWebServiceQueue` and `@AsyncWebServiceResponseQueue`. These new attributes, listed below, enable you to configure the initial and maximum sizes of the Message-driven bean (MDB) pool size, respectively:
  - `messageProcessorInitialPoolSize`
  - `messageProcessorMaxPoolSize`

  For more information, refer to the following topics in "Annotation Reference" in *Developer's Guide for Oracle Infrastructure Web Services*:
  - @AsyncWebServiceQueue Annotation
  - @AsyncWebServiceResponseQueue Annotation
- Enhanced diagnostic and troubleshooting documentation to include additional information about diagnosing common problems with Oracle WSM and policy attachment issues using WLST. For more information, see "Diagnosing Problems" on page 16-1.
- Enhanced message protection keystore configuration documentation. For more information, see the following topics:
  - "Understanding Keys and Certificates" on page 10-2
  - "Configuring Keystores for Message Protection" on page 10-9
  - "Configuring the Credential Store" on page 10-18
- Reorganized documentation describing configuration overrides. For more information, see the following topics:
  - "Attaching Web Service Policies Permitting Overrides" on page 8-25
  - "Attaching Client Policies Permitting Overrides" on page 8-31, specifically a new section "Attaching Client Policies Permitting Overrides Using WLST" on page 8-35.
- Added documentation that describes how to modify a default users group or role to ensure they have the proper permissions to access the Policy Manager. For more

information, see "Modify the User's Group or Role" on page 14-38.

11*g* Release 1 (11.1.1.4)

11*g* Release 1 (11.1.1.4) includes the following new features:

Global Policy Attachments

Oracle Infrastructure Web services provide the ability to create and attach policy sets to subjects on a global scope:

- For conceptual information about policy sets, see "Attaching Policies Globally Using Policy Sets" on page 3-6.
- For information on configuring and managing policy sets using Oracle Enterprise Manager Fusion Middleware Control, see "Creating and Managing Policy Sets" on page 9-1.
- For information on configuring and managing policy sets using WLST, see "Web Services Custom WLST Commands" in the *WebLogic Scripting Tool Command Reference*.
- For information on importing and exporting policy sets using WLST, see "Importing and Exporting Documents in the Repository" on page 17-3.

Oracle Web Services Manager and Oracle Infrastructure Web Services supported on IBM WebSphere

Differences in behavior, and any limitations, are described in "Managing Web Services on IBM WebSphere" in the *Oracle Fusion Middleware Third-Party Application Server Guide*.

SAML 2.0 Support

There is new configuration control for overriding policy attachments and new predefined SAML 2.0 policies.

- A new SAML 2.0 Login Module has been added. See "Configuring the SAML and Kerberos Login Modules" on page 10-60.
- New predefined SAML 2.0 policies have been added. See "Predefined Assertion Templates" on page C-1.

Client-side WS-Trust Support

Support for WS-Trust 1.3 policies has been added. WS-Trust extensions provide methods for issuing, renewing, and validating security tokens. See "WS-Trust Policies and Configuration Steps" on page 10-98.

- A new Automatic Policy Configuration feature dynamically generates the information about an STS config policy by parsing the STS WSDL document. See "Setting Up Automatic Policy Configuration for STS" on page 10-104.
- New predefined WS-Trust assertions have been added. See "Predefined Assertion Templates" on page C-1.

Hardware Token Support

Oracle WSM provides the ability to use the LunaSA Hardware Security Manager (HSM) for key storage. See "Using Hardware Security Modules With Oracle WSM" on page 10-47.

Oracle WebLogic Web Services Monitoring Enhancements

The Web Service Endpoint page in Oracle Enterprise Manager Fusion Middleware Control provides the ability to monitor policy violations for WebLogic JAX-WS Web services. In addition, the tab that displays Oracle WSM policy information has been renamed to **OWSM Policies**. For WebLogic JAX-RPC Web services, the endpoint tab is labeled **WebLogic Policy Violations**.

For more information on monitoring Web services, see "Monitoring the Performance of Web Services" on page 13-1.

Usage Analysis Enhancements

The Usage Analysis page in Oracle Enterprise Manager Fusion Middleware Control provides:

- The option to filter the Policy Subject List by subject type.
- The option to view the available policy subjects in the entire enterprise or only in the local domain/cell.
- The total number of policy subjects to which the policy is attached in the Attachment Count field.

For more information on policy usage analysis, see "Analyzing Policy Usage" on page 7-26.

Test Web Service Enhancements

The Request/Response tabs on Test Web Services page in Oracle Enterprise Manager Fusion Middleware Control have enhanced usability, as follows:

- The Request tab sections are now collapsed by default.
- On the Response tab, the Test Status results has better readability and the composite test results are now highlighted.

For more information on testing Web services, see "Testing Web Services" on page 12-1.

Install Oracle WSM on a Standalone WebLogic Server

If you have a standalone WebLogic Server environment with JAX-WS Web services and clients deployed, you can install Oracle WSM and use it to secure your Web services and clients. For more information, see "Installing Oracle WSM on WebLogic Server" on page 1-7.

Enhanced Specification Support for WS-Policy 1.5 and WS-SecurityPolicy 1.2, 1.3

Supported versions, with links to the specifications, are provided in "Supported Standards" in *Developer's Guide for Oracle Infrastructure Web Services*.

For information about valid version combinations, see "Policy Advertisement" on page 7-28.

New Extensibility Guide for Creating Custom Assertions

All information related to developing custom assertions has been moved from this guide and into the new *Extensibility Guide for Oracle Web Services Manager*.

11*g* Release 1 (11.1.1.3)

11*g* Release 1 (11.1.1.3) includes the following new features:

- Oracle WSM policy attachment to WebLogic Java EE endpoints using Oracle Enterprise Manager Fusion Middleware Control
- Deployment descriptor migration for ADF Business Connect and WebCenter applications using the WebLogic Scripting Tool (WLST)
- Cross-domain policy management of Oracle WSM Policies
- Advertise policies for WebLogic JAX-WS Web services secured with Oracle WSM security policies
- Web services atomic transaction support for SOA Web services and references and WebLogic JAX-WS Web services
- Ability to configure a remote policy store at design time in JDeveloper. For more information, see "Using a Different Oracle WSM Policy Store" in "Developing with Web Services" in the JDeveloper Online Help.
- Shared policy store for Oracle Infrastructure Web services and WebLogic Web services. For information about managing policies in the shared policy store, see "Using Custom Web Service Policies" in "Developing with Web Services" in the JDeveloper Online Help.
- Ability to register Web service sources and to publish registered Web services to UDDI
- Support for the DB2 database in the MDS repository
- Ability to attach policies to Oracle Infrastructure Web Service providers
- Ability to view assertion details for a policy when attaching to an endpoint
- Ability to include a timestamp property for assertion templates that define Transport Security (SSL)
- Ability to manually configure WebLogic Web service repository retrieval properties in Oracle Enterprise Manager Fusion Middleware Control

11g Release 1 (11.1.1.2)

11*g* Release 1 (11.1.1.2) includes the following new features:

- Enhanced administration and policy management for asynchronous Web services
- Ability to define policy alternatives (OR groups)
- Service-side policy configuration overrides
- Oracle WSM policy attachment using the WebLogic Scripting Tool (WLST)
- Ability to upgrade the Oracle WSM policies in the Oracle WSM Repository using WLST commands
- Service identity certification extension for Web services that implement a message-protection policy. The Web service's public certificate is published in the WSDL, and it is no longer necessary for the Web service client to store the Web service's public certificate in its domain-level keystore.
- Enhanced support for permission-based authorization using the oracle.wsm.security.WSFunctionPermission permission check class. In this release, the resource target of the WSFunctionPermission is enhanced to include the actual Web service operation name.
- Ability to browse WSIL documents and import UDDI v3 registries using Fusion Middleware Control, and register services accordingly

- Compliance with WSI-Basic Security Profile
- Support for testing RESTful Web services in Fusion Middleware Control Test Web Service page
- Support for Microsoft SQL Server in the MDS repository
- Ability to use the same Oracle WSM Repository to manage policies across multiple domains. In previous releases, a repository could only be used by a single domain.
- New document, *Oracle Fusion Middleware Interoperability Guide for Oracle Web Services Manager*, that contains the interoperability content previously provided in this document
- Interoperability is certified between Oracle Web Services Manager and Axis 1.4 and WSS4J 1.58 security environments

11g Release 1 (11.1.1)

11*g* Release 1 (11.1.1.9) includes the following new features:

- Integration with the Oracle Fusion Middleware framework
- Shared authorization and authentication infrastructure for Web applications and Web services through Oracle Platform Security Services
- Automatic identity propagation
- Integrated configuration, management, and monitoring of Web services using Oracle Enterprise Manager Fusion Middleware Control
- Use of the Oracle Metadata Repository via Oracle Enterprise Manager Fusion Middleware Control
- Integrated security management and monitoring of WebLogic Web services
- Integrated policy attachment and monitoring support for WebLogic Web services
- Enhanced support for Web services security standards
- Enterprise policy framework with full standards support (WS-Policy, WS-SecurityPolicy, and WS-PolicyAttachment)
- Run Time Services Oriented Architecture (SOA) governance support through reusable run-time policies and bulk attachment of policies
- Policy usage and impact analysis

Part I

Introduction

Part I contains the following chapters:

- Chapter 1, "Overview of Web Services Security and Administration"
- Chapter 2, "Understanding Web Services Security Concepts"
- Chapter 3, "Understanding Oracle WSM Policy Framework"
- Chapter 4, "Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware"

1

Overview of Web Services Security and Administration

Companies worldwide are actively deploying service-oriented architectures (SOA) using Web services, both in intranet and internet environments. While Web services offer many advantages over traditional alternatives (for example, distributed objects or custom software), deploying networks of interconnected Web services still presents key challenges, particularly in terms of security and administration.

This chapter provides an overview of Web services security and administration in Oracle Fusion Middleware 11*g*.

- Web Services Security and Administration in Oracle Fusion Middleware 11g
- Web Service Security and Administration Tasks
- Securing and Administering Oracle Infrastructure Web Services
- Securing and Administering WebLogic Web Services
- Accessing the Security and Administration Tools
- Installing Oracle WSM on WebLogic Server

> **Note:** Oracle Web Services Manager and Oracle Infrastructure Web Services are also supported on IBM WebSphere. Differences in behavior, and any limitations, are described in "Managing Web Services on IBM WebSphere" in *Oracle Fusion Middleware Third-Party Application Server Guide*.

1.1 Web Services Security and Administration in Oracle Fusion Middleware 11*g*

The following highlights the main features of Oracle Fusion Middleware 11g Release 1 (11.1.1):

- **Oracle Web Services Manager (WSM) security and management has been completely redesigned and rearchitected.** The previous release, Oracle WSM 10*g*, was delivered as a standalone product or as a component of the Oracle SOA Suite. In the 11*g* release, Oracle WSM has been integrated into the Oracle WebLogic Server. For complete details, see "Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware" on page 4-1.
- **Oracle Web services can be classified into the following categories**:

  - WebLogic (Java EE) Web services (see "Securing and Administering WebLogic Web Services" on page 1-4)
  - Oracle Infrastructure Web services—SOA, ADF, and WebCenter services (see "Securing and Administering Oracle Infrastructure Web Services" on page 1-3)

  For more information about the two Web service categories and the types of Web services and clients in Oracle Fusion Middleware 11*g*, see *Introducing Web Services*.

- **To support the two categories, there are two types of policies that can be attached to Web services, as defined in the following table.**

Table 1–1 Types of Web Service Policies

Type of Policy	Description
Oracle Web Services Manager (WSM) Policy	Policy provided by the Oracle WSM. You can attach Oracle WSM policies to SOA, ADF, and WebCenter Web services. You can attach Oracle WSM security policies only to WebLogic JAX-WS Web services to interface with the SOA/ADF/WebCenter Web services, for example. (You cannot attach Oracle WSM policies to JAX-RPC Web services.) You manage Oracle WSM policies from Oracle Enterprise Manager Fusion Middleware Control and from the command line using custom WebLogic Scripting Tool (WLST) commands.
WebLogic Web Service Policy	Policy provided by WebLogic Server. For more information about the WebLogic Web service policies, see *Securing WebLogic Web Services for Oracle WebLogic Server*. A subset of WebLogic Web service policies interoperate with Oracle WSM policies. For more information, see "Interoperability with Oracle WebLogic Server 11g Web Service Security Environments" in *Interoperability Guide for Oracle Web Services Manager*. You manage WebLogic Web service policies from WebLogic Administration Console.

- **Application developers can use Oracle JDeveloper to leverage the security and management features of the Oracle WSM policy framework.** For more information about attaching policies using Oracle JDeveloper, see the following sections:
  - "Attaching Policies to Binding Components and Service Components" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.
  - "Securing Web Service Data Controls" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.
  - "Using Oracle Web Services Manager Security Policies" in *Securing WebLogic Web Services for Oracle WebLogic Server*
  - "Using Policies with Web Services" in the "Developing with Web Services" section of the Oracle JDeveloper online help
- **System administrators can use the following tools to secure and administer Web services**:
  - Oracle Enterprise Manager Fusion Middleware Control to secure and administer Oracle Infrastructure Web services, and to secure and test WebLogic (Java EE) Web services.
  - Oracle WebLogic Administration Console to secure and administer WebLogic (Java EE) Web services.

  - Oracle WebLogic Scripting Tool (WLST) to view, configure, and secure SOA, ADF, and WebCenter Web services.

1.2 Web Service Security and Administration Tasks

The following list provides an example of the tasks required to secure and administer Web services:

- Deploy, configure, test, and monitor Web services.
- Enable, publish, and register Web services.
- Directly attach policies to policy subjects to secure and manage Web services and analyze policy usage.
- Attach policies on a global scope to a range of subjects of the same type to secure and manage Web services. Supported scopes include domain, application, partition, module, SOA composite, service, SOA reference, port, and component.
- Create new policies and assertion templates, and manage and configure existing policies.
- Create custom assertions to meet the requirements of your application.
- Manage policy lifecycle to transition from a test to production environment.
- Manage your file-based and database stores in your development and production environments, respectively.
- Test interoperability with other Web services.
- Diagnose problems.

The steps to develop, secure, and administer Web services vary based on the Web service category in use. The following sections outline the steps required:

- Securing and Administering Oracle Infrastructure Web Services
- Securing and Administering WebLogic Web Services

1.3 Securing and Administering Oracle Infrastructure Web Services

To secure and administer Oracle Infrastructure Web services:

- At development time, application developers can attach policies, using Oracle JDeveloper or other IDE, to leverage the security and management features of the Oracle WSM policy framework. For more information about attaching policies using Oracle JDeveloper, see the following sections:
  - "Attaching Policies to Binding Components and Service Components" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.
  - "Securing Web Service Data Controls" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.
  - "Using Policies with Web Services" in the "Developing with Web Services" section of the Oracle JDeveloper online help.
- System administrators can use the tools described in Table 1–2 to secure and administer Oracle Infrastructure Web services.

Table 1–2 Tools Used to Secure and Administer Oracle Infrastructure Web Services

Use this tool...	To...
Oracle Enterprise Manager Fusion Middleware Control	Secure and administer SOA, ADF, and WebCenter services, performing the tasks described in "Web Service Security and Administration Tasks" on page 1-3. To access Oracle Enterprise Manager Fusion Middleware Control, see "Accessing Oracle Enterprise Manager Fusion Middleware Control" on page 1-5. Oracle Enterprise Manager Fusion Middleware Control leverages Oracle Web Services Manager (WSM) to centrally define security and management policies, and enforce them locally at run time. For more information about Oracle WSM, see "Understanding Oracle WSM Policy Framework" on page 3-1. For more information about Oracle Enterprise Manager Fusion Middleware Control, see "Getting Started Using Oracle Enterprise Manager Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide*.
WebLogic Scripting Tool (WLST)	Perform Web service configuration and policy management tasks. To access WLST, see "Accessing the Web Services Custom WLST Commands" on page 1-6. For more information about using WLST, see "Getting Started Using the Oracle WebLogic Scripting Tool (WLST)" in *Oracle Fusion Middleware Administrator's Guide*.

Part II, "Basic Administration" and Part III, "Advanced Administration" describe how to secure and administer SOA, ADF, and WebCenter services in detail.

1.4 Securing and Administering WebLogic Web Services

To secure and administer WebLogic Web services:

- At development time, application developers can attach security policies using Oracle JDeveloper or other IDE. For more information, see the following topics:
  - "Using Policies with Web Services" in the "Developing with Web Services" section of the Oracle JDeveloper online help.
  - "Using Oracle Web Services Manager Security Policies" in *Securing WebLogic Web Services for Oracle WebLogic Server*
- System administrators can use the tools defined in Table 1–3 to secure and administer WebLogic Web services.

Table 1–3 Tools Used to Secure and Administer WebLogic Web Services

Use this tool . . .	To perform the following tasks . . .
Oracle Enterprise Manager Fusion Middleware Control	Leverage Oracle WSM to perform the following tasks: ■ Enforce policies at run time. ■ Manage Oracle WSM security policies and attach to WebLogic Java EE Web services (not clients). ■ Advertise policies for JAX-WS WebLogic Web services secured with Oracle WSM security policies. ■ Test the WebLogic Web service. ■ Monitor the run-time performance of WebLogic Web services. For more information about Oracle WSM, see "Understanding Oracle WSM Policy Framework" on page 3-1. To access Oracle Enterprise Manager Fusion Middleware Control, see "Accessing Oracle Enterprise Manager Fusion Middleware Control" on page 1-5. For more information about Oracle Enterprise Manager Fusion Middleware Control, see "Getting Started Using Oracle Enterprise Manager Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide*. **Note**: The following features are *not supported* for WebLogic Web services in the 11*g* release: ■ WS-SecureConversation, WS-Trust, MTOM, WS-Addressing, WS-ReliableMessaging, or WS-AtomicTransaction using Oracle WSM policies. ■ Security and administration of JAX-RPC WebLogic Web services.
Oracle WebLogic Server Administration Console	Secure and manage WebLogic Web services. To access the Oracle WebLogic Server Administration Console, see "Accessing Oracle WebLogic Administration Console" on page 1-6. For more information about using the Oracle WebLogic Server Administration Console to secure and administer WebLogic Web services, see "Web Services" in the *Oracle WebLogic Server Administration Console Help*.

Part IV, "WebLogic Web Service Administration" provides a roadmap for securing and administering WebLogic Web services.

1.5 Accessing the Security and Administration Tools

The following sections describe how to access the security and administration tools described in the previous sections.

1.5.1 Accessing Oracle Enterprise Manager Fusion Middleware Control

To access Oracle Enterprise Manager Fusion Middleware Control:

1. Start the Oracle WebLogic Server instance.

   For more information, see "Start and stop servers" in the *Oracle WebLogic Server Administration Console Help*.

2. Open a supported Web browser and navigate to the following URL:

   ```
   http://hostname:port/em
   ```

   The Login page displays.

3. Enter the username and password.

The default user name for the administrator user is `weblogic`. This is the account you can use to log in to Fusion Middleware Control for the first time. The password is the one you supplied during the installation of Oracle Fusion Middleware.

4. Click **Login**.

For more information, see "Getting Started Using Oracle Enterprise Manager Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide*.

1.5.2 Accessing Oracle WebLogic Administration Console

To access Oracle WebLogic Administration Console:

1. Start the Oracle WebLogic Server.

   For more information, see "Start and stop servers" in the *Oracle WebLogic Server Administration Console Help*.

2. Open a supported Web browser and navigate to one of the following URLs:

   ```
   http://hostname:port/console
   https://hostname:port/console
   ```

   `hostname` specifies the DNS name or IP address of the Oracle WebLogic Administration Server and *`port`* specifies the address of the port on which the Oracle WebLogic Administration Server is listening for requests (7001 by default).

   Use `https` if you started the Oracle WebLogic Server using the Secure Sockets Layer (SSL).

   For a list of supported browsers, see System Requirements and Supported Platforms for Oracle WebLogic Server at: `http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html`.

   The Login page displays.

3. Enter the username and password.

   You may have specified the username and password during the installation process. This may be the same username and password that you use to start the Oracle Administration Server. Or, a username that is granted one of the default global security roles.

4. Click **Log In**.

For more information, see "Start the Console" in the *Oracle WebLogic Server Administration Console Help*.

1.5.3 Accessing the Web Services Custom WLST Commands

To access the Web services WLST commands:

1. Go to the Oracle Common home directory for your installation, for example `/home/Oracle/Middleware/oracle_common`.

   For information about the Oracle Common home directory and installing Oracle Fusion Middleware, see the *Oracle Fusion Middleware Installation Planning Guide*.

2. Start WLST using the `WLST.sh/cmd` command located in the `oracle_common/common/bin` directory. For example:

   - `/home/Oracle/Middleware/oracle_common/common/bin/wlst.sh` (UNIX)

- `C:\Oracle\Middleware\oracle_common\common\bin\wlst.cmd` (Windows)

  When executed, these commands start WLST in offline mode. To use the Web services WLST commands, you must use WLST in online mode.

3. Start Oracle WebLogic Server.

   For more information, see "Start and stop servers" in the *Oracle WebLogic Server Administration Console Help*.

4. Connect to the running WebLogic Server instance using the `connect()` command. For example, the following command connects WLST to the Admin Server at the URL `myAdminServer.example.com:7001` using the username/password credentials `weblogic/`*`password`*:

   ```
   connect("weblogic","password","t3://myAdminServer.example.com:7001")
   ```

For more information about using WLST, see "Using the WebLogic Scripting Tool" in *Oracle WebLogic Scripting Tool*.

For more information about the Web Services WLST commands, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

1.6 Installing Oracle WSM on WebLogic Server

Oracle WSM is installed by default when you install Oracle Fusion Middleware SOA Suite or Oracle Application Development Runtime. However, if you have a standalone WebLogic Server environment with JAX-WS Web services and clients deployed, you can install Oracle WSM and use it to secure your Web services and clients.

> **Note:** Oracle WSM is licensed only through SOA Suite; a standalone license is not available. To secure Web service clients and services using Oracle WSM on base Weblogic Server, you must acquire a SOA Suite license in addition to a Weblogic Server license.

To use Oracle WSM with WebLogic Server, you need Java Required Files (JRF) and Oracle Enterprise Manager Fusion Middleware Control. JRF consists of those components, such as Oracle WSM, that provide common functionality for Oracle business applications and application frameworks. Oracle Enterprise Manager Fusion Middleware Control is used to secure and administer WebLogic Web services.

Neither JRF or Fusion Middleware Control are included in the WebLogic Server installation. The following procedure describes the steps required to install and configure Oracle WSM with WebLogic Server.

1. Prepare for the installation by reviewing the concepts and requirements as described in the *Oracle Fusion Middleware Installation Planning Guide*.
2. Download the following Oracle Fusion Middleware software components:

   - Oracle WebLogic Server
   - Oracle Application Development Runtime
   - Oracle Fusion Middleware Repository Creation Utility

   For download sites, see "Obtain the Oracle Fusion Middleware Software" in *Oracle Fusion Middleware Installation Planning Guide*.
3. Create the MDS schema in your database.

Oracle Application Developer includes Oracle WSM Policy Manager and Oracle WSM-PM Extension. These components require that the MDS schema exists in your database prior to installation. You must run the Repository Creation Utility (RCU) to create the MDS schema in your database. For instructions, see "Creating Schemas" in *Oracle Fusion Middleware Repository Creation Utility User's Guide*.

> **Note:** In the Select Components screen, be sure to select **Metadata Services** under **AS Common Schemas**.

4. Install WebLogic Server. For detailed instructions, see *Installation Guide for Oracle WebLogic Server*.

   Be sure to take note of the location that you specify for the Middleware Home directory as you will need to provide it during the Application Developer installation.

5. Install Application Developer. For detailed instructions, see "Installation Instructions" in *Oracle Fusion Middleware Installation Guide for Application Developer*.

   > **Note:** In the Specify Installation Location screen, specify the Middleware home location that you provided during the WebLogic Server installation.

6. Create a domain that includes Oracle Enterprise Manager, Oracle WSM, and JRF using the Configuration Wizard. For details, see "Configuring Application Developer" in *Oracle Fusion Middleware Installation Guide for Application Developer*.

   > **Note:** In the Select Domain Source screen of the Configuration Wizard, select **Oracle Enterprise Manager** and **Oracle WSM Policy Manager**. **Oracle JRF** is automatically selected as a dependency.

You can now secure and administer WebLogic Web services as described in "Securing and Administering WebLogic Web Services" on page 1-4.

2 Understanding Web Services Security Concepts

This chapter introduces the Web services security concepts. It is divided into the following sections:

- Securing Web Services
- How Oracle Fusion Middleware Secures Web Services and Clients

For an introduction to general Web service concepts, see "What are Web Services" in *Introducing Web Services*.

2.1 Securing Web Services

Because of its nature (loosely coupled connections) and its use of open access (mainly HTTP), SOA implemented by Web services adds a new set of requirements to the security landscape. Web services security includes several aspects:

- **Authentication**—Verifying that the user is who she claims to be. A user's identity is verified based on the credentials presented by that user, such as:

  1. Something one has, for example, credentials issued by a trusted authority such as a passport (real world) or a smart card (IT world).
  2. Something one knows, for example, a shared secret such as a password.
  3. Something one is, for example, biometric information.

  Using a combination of several types of credentials is referred to as "strong" authentication, for example using an ATM card (something one has) with a PIN or password (something one knows).

- **Authorization (or Access Control)**—Granting access to specific resources based on an authenticated user's entitlements. Entitlements are defined by one or several attributes. An attribute is the property or characteristic of a user, for example, if "Marc" is the user, "conference speaker" is the attribute.
- **Confidentiality, privacy**—Keeping information secret. Accesses a message, for example a Web service request or an email, as well as the identity of the sending and receiving parties in a confidential manner. Confidentiality and privacy can be achieved by encrypting the content of a message and obfuscating the sending and receiving parties' identities.
- **Integrity, non repudiation**—Making sure that a message remains unaltered during transit by having the sender digitally sign the message. A digital signature is used to validate the signature and provides non-repudiation. The timestamp in the signature prevents anyone from replaying this message after the expiration.

Web services security requirements also involve credential mediation (exchanging security tokens in a trusted environment), and service capabilities and constraints (defining what a Web service can do, under what circumstances).

In many cases, Web services security tools such as Oracle WSM rely on Public Key Infrastructure (PKI) environments. A PKI uses cryptographic keys (mathematical functions used to encrypt or decrypt data). Keys can be private or public. In an asymmetric cipher model, the receiving party's public key is used to encrypt plaintext, and the receiving party's matching private key is used to decrypt the ciphertext. Also, a private key is used to create a digital signature by signing the message, and the public key is used for verifying the signature. Public-key certificates (or certificates, for short) are used to guarantee the integrity of public keys.

Web services security requirements are supported by industry standards both at the transport level (Secure Socket Layer) and at the application level relying on XML frameworks.

For more information about the specifications, standards, and security tokens supported by Web services, see Appendix A, "Web Service Security Standards."

Note: Oracle has been instrumental in contributing to emerging standards, in particular the specifications hosted by the OASIS Web Services Secure Exchange technical committee.

2.1.1 Transport-level Security

Secure Socket Layer (SSL), otherwise known as Transport Layer Security (TLS), the Internet Engineering Task Force (IETF) officially standardized version of SSL, is the most widely used transport-level data-communication protocol providing:

- Authentication (the communication is established between two trusted parties).
- Confidentiality (the data exchanged is encrypted).
- Message integrity (the data is checked for possible corruption).
- Secure key exchange between client and server.

SSL provides a secure communication channel, however, when the data is not "in transit," the data is not protected. This makes the environment vulnerable to attacks in multi-step transactions. (SSL provides point-to-point security, as opposed to end-to-end security.)

2.1.2 Application-level Security

Application-level security complements transport-level security. Application-level security is based on XML frameworks defining confidentiality, integrity, authenticity; message structure; trust management and federation.

Data confidentiality is implemented by XML Encryption. XML Encryption defines how digital content is encrypted and decrypted, how the encryption key information is passed to a recipient, and how encrypted data is identified to facilitate decryption.

Data integrity and authenticity are implemented by XML Signature. XML Signature binds the sender's identity (or "signing entity") to an XML document. Signing and signature verification can be done using asymmetric or symmetric keys.

Signature ensures non-repudiation of the signing entity and proves that messages have not been altered since they were signed. Message structure and message security are implemented by SOAP and its security extension, WS-Security. WS-Security defines how to attach XML Signature and XML Encryption headers to SOAP messages. In addition, WS-Security provides profiles for 5 security tokens: Username (with password digest), X.509 certificate, Kerberos ticket, Security Assertion Markup Language (SAML) assertion, and REL (rights markup) document.

The SOAP envelope body includes the business payload, for example a purchase order, a financial document, or simply a call to another Web service. SAML is one of the most interesting security tokens because it supports both authentication and authorization. SAML is an open framework for sharing security information on the Internet through XML documents. SAML includes 3 parts:

- SAML Assertion—How you define authentication and authorization information.
- SAML Protocol—How you ask (SAML Request) and get (SAML Response) the assertions you need.
- SAML Bindings and Profiles—How SAML assertions ride "on" (Bindings) and "in" (Profiles) industry-standard transport and messaging frameworks.

The full SAML specification is used in browser-based federation cases. However, web services security systems such as Oracle WSM only use SAML assertions. The protocol and bindings are taken care of by WS-Security and the transport protocol, for example HTTP.

SAML assertions and references to assertion identifiers are contained in the WS-Security Header element, which in turn is included in the SOAP Envelope Header element (described in the WS-Security SAML Token Profile). The SAML security token is particularly relevant in situations where identity propagation is essential.

2.1.3 Web Service Security Requirements

The following summarize the Web service security requirements:

1. Transport-level security to protect the communication channel between the Web service consumer and the Web service provider, with transport-level authentication by requiring username, SAML, or JWT tokens.
2. Transport-level security with message-level authentication by requiring username or SAML tokens.
3. Message-level security to ensure confidentiality by digitally encrypting message parts; integrity using digital signatures; and authentication by requiring username, X.509, or SAML tokens.

Oracle Web Services Manager (WSM) is designed to define and implement Web services security in heterogeneous environments, including authentication, authorization, message encryption and decryption, signature generation and validation, and identity propagation across multiple Web services used to complete a single transaction.

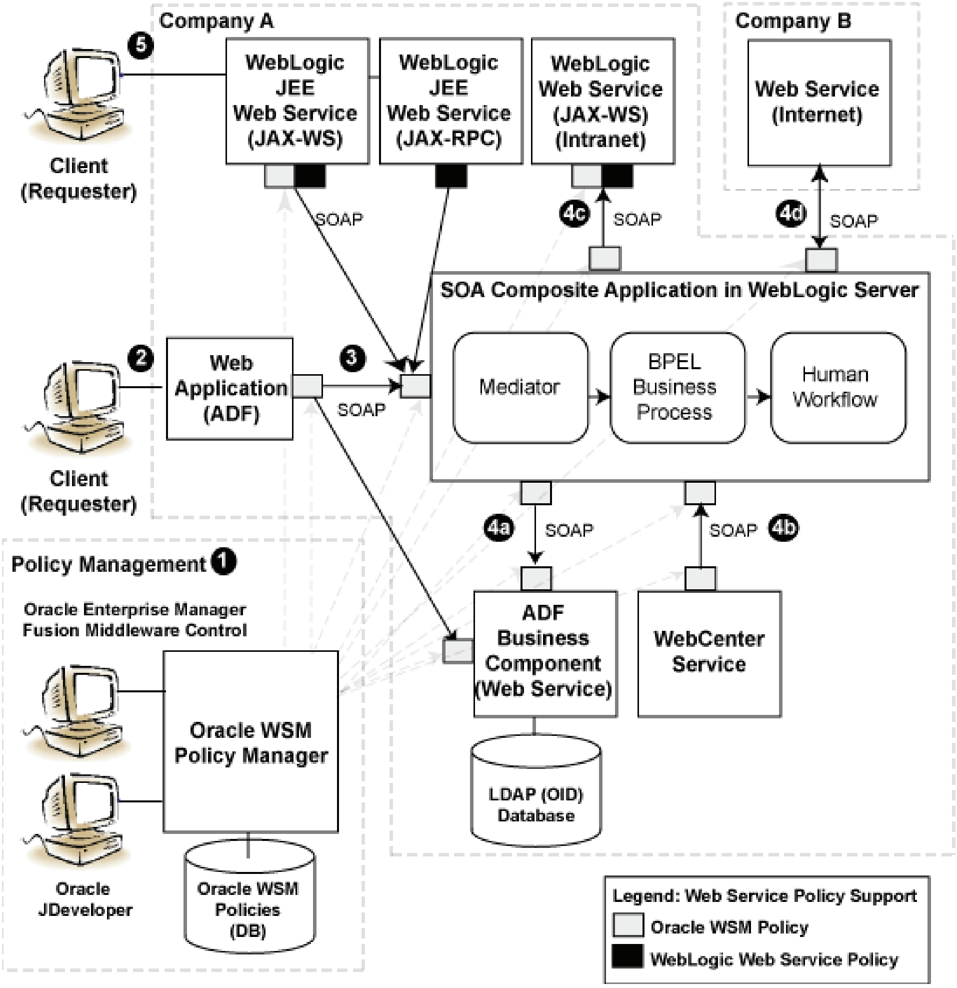
2.2 How Oracle Fusion Middleware Secures Web Services and Clients

Figure 2–1 shows an Oracle Fusion Middleware application that demonstrates some common interactions between Web services and their clients. How security is managed at each step in the process is explained following the figure.

The Oracle WSM Policy Manager (labeled as OWSM in Figure 2–1) is the security linchpin for Oracle Fusion Middleware Web services and SOA applications. For more information about how the Oracle WSM Policy Manager manages the policy framework, see Chapter 3, "Understanding Oracle WSM Policy Framework."

> **Note:** To ensure the security of your Web services, the Policy Manager must be running before starting any Web services. If the Policy Manager is deployed on a separate server (other than the servers running Web services), make sure that it is up and running before starting the other servers.

Figure 2–1 Example of Oracle Fusion Middleware Application



Company A
5
WebLogic JEE Web Service (JAX-WS)
WebLogic JEE Web Service (JAX-RPC)
WebLogic Web Service (JAX-WS) (Intranet)
Company B
Web Service (Internet)
Client (Requester)
SOAP
4c
SOAP
4d
SOAP
SOA Composite Application in WebLogic Server
2
Web Application (ADF)
3
SOAP
Mediator
BPEL Business Process
Human Workflow
Client (Requester)
4a
SOAP
SOAP
4b
Policy Management
1
Oracle Enterprise Manager Fusion Middleware Control
Oracle WSM Policy Manager
ADF Business Component (Web Service)
WebCenter Service
LDAP (OID) Database
Oracle JDeveloper
Oracle WSM Policies (DB)
Legend: Web Service Policy Support
□ Oracle WSM Policy
■ WebLogic Web Service Policy

As shown in the previous figure, there are two types of policies that can be attached to Web services: Oracle WSM policies and WebLogic Server polices. For more information, see Table 1–1, " Types of Web Service Policies".

The following describes in more detail the Web service and client interactions called out in the previous figure, and how security is managed at each step in the process. As noted in the figure, security is managed using both Oracle WSM policies and WebLogic Web service policies.

1. At design time, you attach Oracle WSM and WebLogic Web service policies to applications programmatically using your favorite IDE, such as Oracle JDeveloper.

   Alternatively, at deployment time you attach policies to SOA composites, ADF, and WebCenter applications using the Oracle Enterprise Manager Fusion Middleware Control, and to WebLogic Web services (Java EE) using the WebLogic Server Administration Console (not shown in the figure).

Note: Policies that are attached to WebLogic Web services at design time cannot be detached at deployment time. You can only attach new policies.

2. A user logs in to the ADF Web application.

   The user may be internal or external to Company A.

3. Using a Web service data control, the ADF Web application accesses a service, such as a WebLogic Web service, a SOA composite application, or an ADF Business Component.

   At the Web service client side, Oracle WSM intercepts the SOAP message request to the service, injects the relevant tokens, and signs and encrypts the message, as required by the attached policies.

   At the Web service side, Oracle WSM intercepts the SOAP message request to the service, extracts the tokens, and verifies the client's credentials against an identity management infrastructure (for example, a file, an LDAP-compliant directory, or Oracle Access Manager), as required by the attached policies.

4. Interactions with the SOA service components (shown in the figure) include:

   a. The SOA service component accesses an ADF Business Component to query or update tables in a database.

   b. A WebCenter client access the SOA service component to process a customer request.

   c. The SOA service component accesses the Web service internal to Company A to accomplish a specific task.

   d. The SOA service component accesses a Web service via an external provider (Company B) to accomplish a specific task. As long as you know the URL that identifies the WSDL document, you can access the Web service.

   Again, at the Web service client side, Oracle WSM intercepts the SOAP message request to the service, injects the relevant tokens, and signs and encrypts the message, as required by the attached policies.

   At the Web service side, Oracle WSM intercepts the SOAP message request to the service, extracts the tokens, and verifies the client's credentials against an identity management infrastructure (for example, a file, an LDAP-compliant directory, or Oracle Access Manager), as required by the attached policies.

5. A client accesses a WebLogic Java EE Web service.

   In this case, components in a larger composite application interact with the WebLogic Web service. An *Oracle WSM* policy is used to secure the WebLogic JAX-WS Web service client. A *WebLogic Web service* policy is used to secure the WebLogic JAX-RPC service client.

3 Understanding Oracle WSM Policy Framework

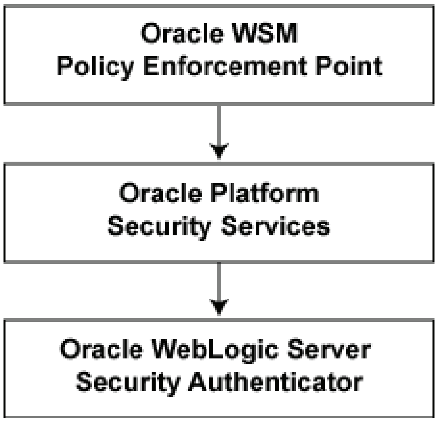
This chapter contains the following sections:

3.1 Overview of Oracle WSM Policy Framework

Oracle Web Services Manager (WSM) provides a policy framework to manage and secure Web services consistently across your organization. Oracle WSM can be used by both developers, at design time, and system administrators in production environments.

The policy framework is built using the WS-Policy standard. The Oracle WSM Policy Enforcement Point (PEP) leverages Oracle Platform Security Service (OPSS) and the Oracle WebLogic Server authenticator for authentication and permission-based authorization, as shown in the following figure.

Figure 3–1 Oracle WSM Policy Framework Leverages OPSS and Oracle WebLogic Server Security



Developers can leverage the Oracle WSM policy framework from Oracle JDeveloper. For more information, see "Developing with Web Services" in the Oracle JDeveloper online help.

System administrators can leverage the Oracle WSM through the Oracle Enterprise Manager Fusion Middleware Control to:

- Centrally define policies using the Oracle WSM Policy Manager.
- Enforce Oracle WSM security and management polices locally at run time.

All of Oracle WSM's functionality is accessible to administrators from Oracle Enterprise Manager Fusion Middleware Control. Part II, "Basic Administration" and Part III, "Advanced Administration" describe the security and administration tasks in more detail.

The following list provides examples of specific tasks that you can perform using Oracle WSM:

- Handle WS-Security (for example, encryption, decryption, signing, signature validation, and so on)
- Define authentication and authorization policies against an LDAP directory.
- Generate standard security tokens (such as SAML and JWT tokens) to propagate identities across multiple Web services used in a single transaction.
- Segment policies into different namespaces by creating policies within different folders.
- Examine log files.

Figure 3–2 shows the main components of Oracle WSM architecture.

Figure 3–2 Components of Oracle WSM Architecture

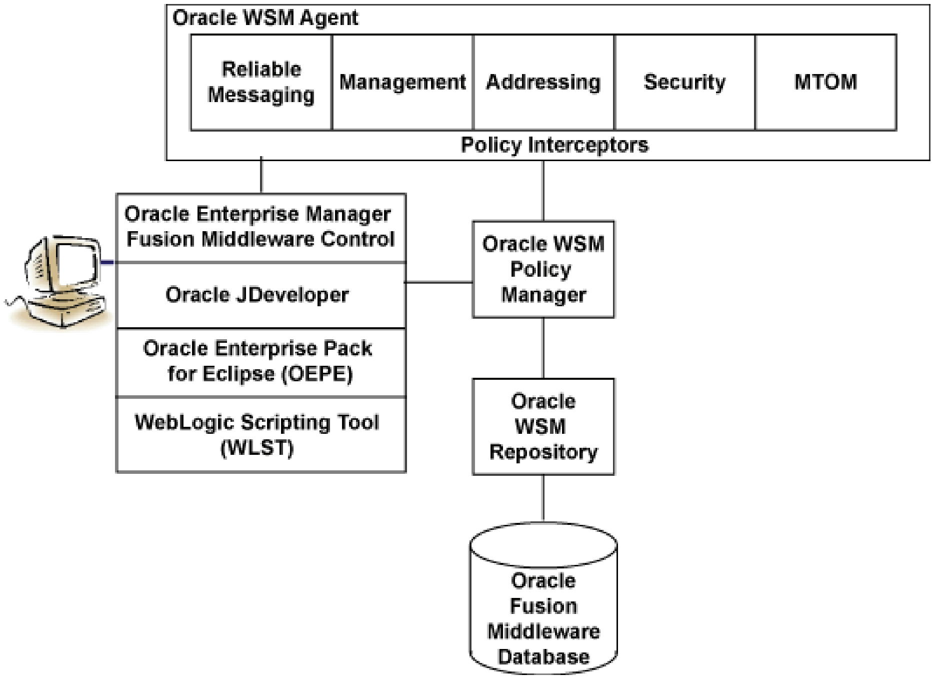

Oracle WSM Agent
Reliable Messaging
Management
Addressing
Security
MTOM
Policy Interceptors
Oracle Enterprise Manager Fusion Middleware Control
Oracle JDeveloper
Oracle Enterprise Pack for Eclipse (OEPE)
WebLogic Scripting Tool (WLST)
Oracle WSM Policy Manager
Oracle WSM Repository
Oracle Fusion Middleware Database

Table 3–1 describes the components of Oracle WSM shown in the previous figure.

Table 3–1 Components of Oracle WSM Architecture

Oracle WSM Component	Description
Oracle Enterprise Manager Fusion Middleware Control	Enables administrators to access Oracle WSM's functionality to manage, secure, and monitor Web services.
Oracle JDeveloper	Provides a full-featured Java IDE for SOA that can be used for end-to-end development of Web services. Using visual and declarative tools, developers can build Oracle SOA, ADF, WebCenter, and WebLogic Java EE Web services, automatically deploy them to an instance of Oracle WebLogic Server, and immediately test the running Web service. Alternatively, JDeveloper can be used to drive the creation of Web services from WSDL descriptions. JDeveloper is Ant-aware. You can use this tool to build and run Ant scripts for assembling the client and for assembling and deploying the service. For more information, see the Oracle JDeveloper online help. For information about installing JDeveloper, see *Oracle Fusion Middleware Installation Guide for Oracle JDeveloper*.
WebLogic Scripting Tool (WSLT)	Enables administrators to view and configure Web services, and manage Web service policies from the command line. For more information, see *WebLogic Scripting Tool Command Reference*.
Oracle WSM Policy Manager	Reads/writes the policies, including predefined and custom policies from the Oracle WSM Repository.
Oracle WSM Agent	Manages the enforcement of policies via the Policy Interceptor Pipeline.

Table 3–1 (Cont.) Components of Oracle WSM Architecture

Oracle WSM Component	Description
Policy Interceptors	Enforce policies, including reliable messaging, management, addressing, security, and Message Transmission Optimization Mechanism (MTOM). For more information, see "How Policies are Executed" on page 3-7.
Oracle WSM Repository	Stores Oracle WSM metadata, such as policies, policy sets, assertions templates, and policy usage data. The Oracle WSM Repository is available as a database (for production use) or as files in the file system (for development use in JDeveloper).
Oracle Fusion Middleware Database	Provides database support for the Oracle WSM Repository.

3.2 What Are Policies?

Policies describe the capabilities and requirements of a Web service such as whether and how a message must be secured, whether and how a message must be delivered reliably, and so on.

Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9) supports the types of policies defined in Table 3–2. The policies are part of the Oracle WSM enterprise policy framework which allows policies to be centrally created and managed.

Table 3–2 Types of Policies

Policy Type	Description
WS-Reliable Messaging	Reliable messaging policies that implement the WS-ReliableMessaging standard describes a wire-level protocol that allows guaranteed delivery of SOAP messages, and can maintain the order of sequence in which a set of messages are delivered. The technology can be used to ensure that messages are delivered in the correct order. If a message is delivered out of order, the receiving system can be configured to guarantee that the messages will be processed in the correct order. The system can also be configured to deliver messages at least once, not more than once, or exactly once. If a message is lost, the sending system re-transmits the message until the receiving system acknowledges it receipt.
Management	Management policies that log request, response, and fault messages to a message log. Management policies may include custom policies.

Table 3–2 (Cont.) Types of Policies

Policy Type	Description
WS-Addressing	WS-Addressing policies that verify that SOAP messages include WS-Addressing headers in conformance with the WS-Addressing specification. Transport-level data is included in the XML message rather than relying on the network-level transport to convey this information.
Security	Security policies that implement the WS-Security 1.0 and 1.1 standards. They enforce message protection (message integrity and message confidentiality), and authentication and authorization of Web service requesters and providers. The following token profiles are supported: username token, X.509 certificate, Kerberos ticket, Security Assertion Markup Language (SAML), and JWT. For more information about Web service security concepts and standards, see "Understanding Web Services Security Concepts" on page 2-1 and "Web Service Security Standards" on page A-1
Message Transmission Optimization Mechanism (MTOM)	Binary content, such as an image in JPEG format, can be passed between the client and the Web service. In order to be passed, the binary content is typically inserted into an XML document as an `xsd:base64Binary` string. Transmitting the binary content in this format greatly increase the size of the message sent over the wire and is expensive in terms of the required processing space and time. Using Message Transmission Optimization Mechanism (MTOM), binary content can be sent as a MIME attachment, which reduces the transmission size on the wire. The binary content is semantically part of the XML document. Attaching an MTOM policy ensures that the message is converted to a MIME attachment before it is sent to the Web service or client.

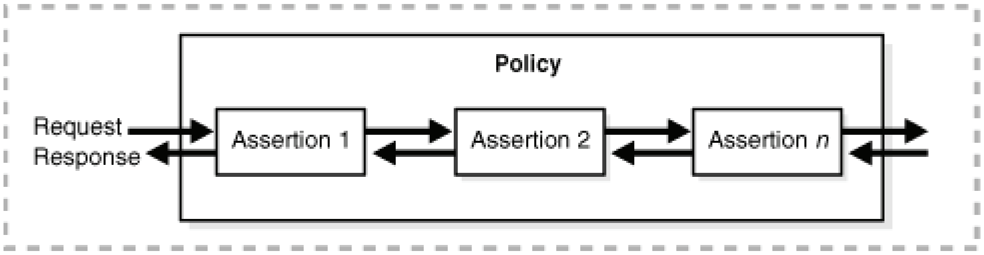
3.3 Building Policies Using Policy Assertions

A policy is comprised of one or more policy **assertions**. A policy assertion is the smallest unit of a policy that performs a specific action for the request and response operations. Assertions, like policies, belong to one of the following categories: Reliable Messaging, Management, WS-Addressing, Security, and MTOM.

Policy assertions are chained together in a pipeline. The assertions in a policy are executed on the request message and the response message, and the same set of assertions are executed on both types of messages. The assertions are executed in the order in which they appear in the pipeline.

Figure 3–3 illustrates a typical execution flow. For the request message, Assertion 1 is executed first, followed by Assertion 2, and Assertion *n*. Although the same assertions may be executed on the response message (if a response is returned at all), the actions performed on the response message differ from the request message, and the assertions are executed on the response message in reverse order. For the response message in Figure 3–3, Assertion *n* is executed first, followed by Assertion 2, then Assertion 1.

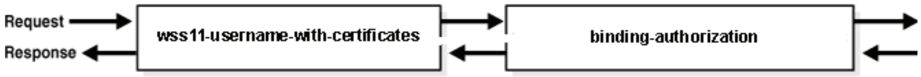
Figure 3–3 Policy Containing Assertions



For example, in Figure 3–4, the policy contains two assertions:

1. wss11-username-with-certificates—Built using the wss11_username_token_with_message_protection_service_template, authenticates the user based on credentials in the WS-Security UsernameToken SOAP header.
2. binding-authorization—Built using the binding_authorization_template, provides simple role-based authorization for the request based on the authenticated subject at the SOAP binding level.

Figure 3–4 Example Policy With Two Assertions



When the request message is sent to the Web service, the assertions are executed in the order shown. When the response message is returned to the client, the same assertions are executed, but this time in reverse order. The behavior of the assertion for the request message differs from the behavior for the response message. And, in some instances, it is possible that nothing happens on the response. For example, in the example above, the authorization assertion is only executed as part of the request.

3.4 Attaching Policies to Subjects

A policy subject is the target resource to which the policies are attached. Policy subjects include Web services endpoints, Web service clients, SOA service endpoints, SOA clients, and SOA components. There are different policies for different types of resources (for example, a Web service or a SOA component).

You can attach one or more policies to a policy subject, either by directly attaching an individual policy to a subject, or using bulk attachment. You can also attach policies globally to a set of subjects by type using policy sets. For more information, see "Attaching Policies Globally Using Policy Sets" on page 3-6. When the policy is attached to a policy subject, enforcement of the policy begins immediately.

If a policy on the client side is modifying the message, for example to encrypt the message, there must be a corresponding policy on the Web service side, for example, to decrypt the policy. Otherwise, the message request will fail.

3.5 Attaching Policies Globally Using Policy Sets

Note: Policy sets are supported for Oracle Infrastructure Web services only.

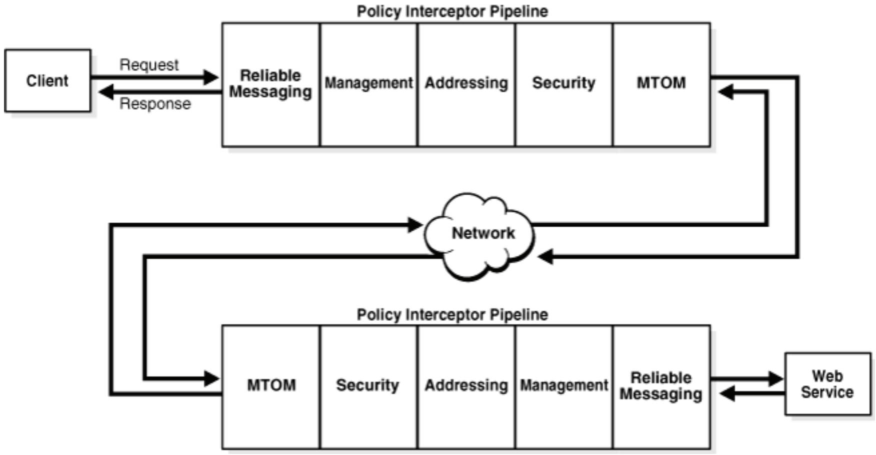
A policy set, which can contain multiple policy references, is an abstract representation that provides a means to attach policies globally to a range of endpoints of the same type. Attaching policies globally using policy sets allows the administrator to ensure that all subjects are secured in situations where the developer, assembler, or deployer did not explicitly specify the policies to be attached. Policies that are attached using a policy set are considered externally attached. For more information about attaching policies globally using policy sets, see Chapter 9, "Creating and Managing Policy Sets."

3.6 How Policies are Executed

When a request is made from a service consumer (also known as a client) to a service provider (also known as a Web service), the request is intercepted by one or more policy interceptors. These interceptors execute policies that are attached to the client and to the Web service. There are five types of interceptors (reliable messaging, management, WS-Addressing, security, and MTOM) that together form a policy interceptor chain. Each interceptor executes policies of the same type. The security interceptor intercepts and executes security policies, the MTOM interceptor intercepts and executes MTOM policies, and so on.

Policies attached to a client or Web service are executed in a specific order via the Policy Interceptor Pipeline, as shown in Figure 3–5.

Figure 3–5 Policy Interceptors Acting on Messages Between a Client and Web Service



As shown in the previous figure, when a client or a Web service *initiates* a message, whether it be a request message in the case of a client, or a response message in the case of a Web service, the policies are intercepted in the following order: Reliable Messaging, Management, Addressing, Security, and MTOM. When a client or a Web service *receives* a message, that is, a request message in the case of the Web service or a response message in the case of a client, the policies are executed in the reverse order: MTOM, Security, Addressing, Management, and Reliable Messaging.

A message may have one or more policies attached. Not every message will contain each type of policy. A message may contain a security policy and an MTOM policy. In this instance, the security interceptor executes the security policy, and the MTOM interceptor executes the MTOM policy. In this example, the other interceptors are not involved in processing the message.

The following describes how the policy interceptors act on messages between the client and the Web service. (Refer to Figure 3–5.)

1. The client sends a request message to a Web service.
2. The policy interceptors intercept and execute the policies attached to the client. After the client policies are successfully executed, the request message is sent to the Web service.
3. The request message is intercepted by policy interceptors which then execute any service policies that are attached to the Web service.

4. After the service policies are successfully executed, the request message is passed to the Web service. The Web service executes the request message and returns a response message.
5. The response message is intercepted by the policy interceptors which execute the service policies attached to the Web service. After the service policies are successfully executed, the response message is sent to the client.
6. The response message is intercepted by the policy interceptors which execute any client policies attached to the client.
7. After the client policies are successfully executed, the response message is passed to the client.

3.7 Oracle WSM Predefined Policies and Assertion Templates

There is a set of predefined policies and assertion templates that are automatically available when you install Oracle Fusion Middleware. The predefined policies are based on common best practice policy patterns used in customer deployments.

You can immediately begin attaching these predefined policies to your Web services or clients. You can configure the predefined policies or create a new policy by making a copy of one of the predefined policies.

Predefined policies are constructed using assertions based on predefined assertion templates. You can create new assertion templates, as required.

For more information about the predefined policies and assertion templates, see:

- "Predefined Policies" on page B-1.
- "Predefined Assertion Templates" on page C-1.

> **Note:** WS-SecurityPolicy defines *scenarios* that describe examples of how to set up WS-SecurityPolicy policies for several security token types described in the WS-Security specification (supporting both WS-Security 1.0 and 1.1). The Oracle WSM predefined policies support a subset of the WS-SecurityPolicy scenarios that represents the most common customer use cases.

3.8 Defining Multiple Policy Alternatives (OR Groups)

To define multiple alternatives for policy enforcement, you can define a set of assertions, called an **OR group**, within a service policy. At run time, based on the assertions defined in the OR group on the service side, a client has the flexibility to choose which *one* of the assertions to enforce.

For example, if a service-side policy defines an OR group that consists of the following assertions:

- wss11-saml-with-certificates
- wss11-username-with-certificates

At run-time, the client can choose to enforce either the wss11-saml-with certificates assertion OR wss11-username-with-certificates assertion.

There is no limit to the number of assertions that can be included in an OR group. For a set of assertions within the OR group, if a request message satisfies the first assertion, then the first assertion gets executed and the response is sent accordingly.

Each assertion should be valid for the policy and support the policy requirements. For example, you should not include a log assertion in an OR group that otherwise contains security assertions and that is designed to enforce security. In this case, the log assertion would pass in the event the security assertions failed, resulting in no security.

When defining the OR group, carefully consider the order in which the assertions are added and the settings that are configured. For example, consider the following scenario:

- On the client side, you have attached the wss11_username_token_with_message_protection_client_policy policy with `Include Timestamp` enabled.
- On the service side, you have attached a custom OR group policy with two `wss11_username_token_with_message_protection_service_template` assertions defined, the first with `Include Timestamp` disabled and the second with `Include Timestamp` enabled.

In this scenario, the first assertion will get executed and the response will be sent with no timestamp. As a result, processing on the client side will fail because it is expecting a timestamp. This type of situation can occur whenever a client policy assertion expects a greater number of security requirements than the executed service policy assertion.

The following predefined service policies contain OR groups:

- oracle/wss_saml_or_username_token_over_ssl_service_policy—For more information, see "oracle/wss_saml_or_username_token_over_ssl_service_policy" on page B-18.
- oracle/wss_saml_or_username_token_service_policy—For more information, see "oracle/wss_saml_or_username_token_service_policy" on page B-8.
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy—For more information, see "oracle/wss11_saml_or_username_token_with_message_protection_service_policy" on page B-32.
- oracle/multi_token_rest_service_policy—For more information, see "oracle/multi_token_rest_service_policy" on page B-5.
- oracle/multi_token_over_ssl_rest_service_policy—For more information, see "oracle/multi_token_over_ssl_rest_service_policy" on page B-16.

3.9 Overriding Security Policy Configuration

Multiple Web services or clients may use the same policy. Each may have different policy configuration requirements such as username and password.

Oracle WSM policy configuration override enables you to update the configuration on a per service or client basis without creating new policies for each. In this way, you can create policies that define default configuration values and customize those values based on your run-time requirements.

For example, you might specify the username and password when configuring a client policy, as the information may vary from client to client.

For more information about overriding security policy configuration, see "Attaching Client Policies Permitting Overrides" on page 8-31 and "Attaching Web Service Policies Permitting Overrides" on page 8-25.

You can define whether a configuration property is overridable when creating custom assertions, as described in "Creating Custom Assertions" in *Extensibility Guide for Oracle Web Services Manager*.

3.10 Recommended Naming Conventions for Policies

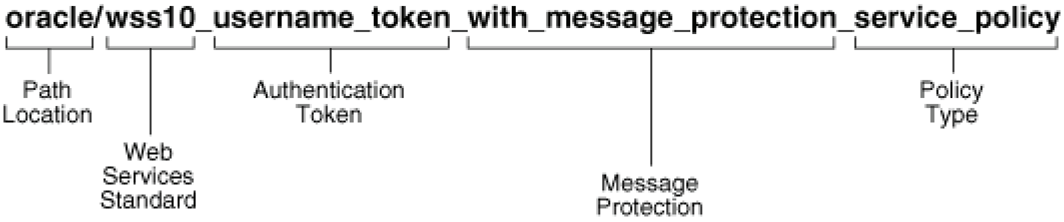
The valid characters for directory, policy, and assertion template names are:

- Uppercase and lowercase letters
- Numerals
- Currency symbol ($)
- Underscore (_)
- Hyphen (-)
- Spaces

Note: The first character in the name cannot be a hyphen or space.

Oracle recommends that you encode as much information as possible into the name of the policy so that you can tell, at a glance, what the policy does. For example, one of the predefined security policies that is delivered with Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9) is named oracle/wss10_username_token_with_message_protection_service_policy. Figure 3–6 identifies the different parts of this predefined policy name.

Figure 3–6 Identifying the Different Parts of a Policy Name



The following convention is used to name the predefined policies. The parts of the policy name are separated with an underscore character (_).

- Path Location – All policies are identified by the directory in which the policy is located. All predefined policies that come with the product are in the `oracle` directory.
- Web services Standard – If the policy uses a WS-Security standard, it is identified with wss10 (WS-Security 1.0) or wss11 (WS-Security 1.1). If the policy is set to wss, it indicates that it is independent of WS-Security 1.0 or 1.1.
- Authentication token – If the policy authenticates users, then the type of token is specified. The predefined options include:
  - http_token – HTTP token
  - jwt_token – JWT token
  - kerberos_token – Kerberos token
  - saml_token – SAML token

  - saml_hok_token –SAML holder of key token
  - saml20_token – SAML 2.0 token
  - saml20_token_bearer –SAML Bearer 2.0 token
  - username_token – Username and password token
  - x509_token – X.509 certificate token

  You can also define custom authentication tokens.

- Transport security – If the policy requires that the message be sent over a secure transport layer, then the token name is followed by *over_ssl,* for example, wss_ http_token*_over_ssl_*client_template.
- Message protection – If the policy also provides message confidentiality and message integrity, then this is indicated using the phrase *with_message_protection* as in Figure 3–6.
- Policy Type – Indicates the type of policy or assertion template— *client* or *service*. Use the term *policy* to indicate that it is a policy, or *template* to indicate that it is an assertion template. For example, there are predefined policy and template assertions that are distinguished, as follows:

  wss10_message_protection_service_policy

  wss10_message_protection_service_template

Whatever conventions you adopt, Oracle recommends you take some time to consider how to name your policies. This will make it easier for you to keep track of your policies as your enterprise grows and you create new policies.

It is recommended that you keep any policies you create in a directory that is separate from the oracle directory where the predefined policies are located. You can organize your policies at the root level, in a directory other than oracle, or in subdirectories. For example, all of the following are valid:

- wss10_message_protection_service_policy
- oracle/hq/wss10_message_protection_service_policy
- hq/wss10_message_protection_service_policy

> **Note:** Use of the prefix "oracle_" in the policy name (for example, oracle_wss_http_token_service_policy) is not recommended as a best practice.

4

Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware

In Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9), Oracle Web Services Manager (WSM) security and management has been completely redesigned and rearchitected. The previous release, Oracle WSM 10*g*, was delivered as a standalone product or as a component of the Oracle SOA Suite. In the 11*g* release, Oracle WSM has been integrated with Oracle WebLogic Server as part of the Oracle Fusion Middleware SOA Suite.

This chapter contains the following sections:

4.1 How Oracle WSM 10*g* is Redesigned in Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9)

Oracle WSM 10*g* has been rearchitected in Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9), as follows:

- **Oracle WSM Agent functionality is integrated into Oracle WebLogic Server.** In Oracle Fusion Middleware 11g, the Oracle WSM 10*g* Agents are managed by the security and management policy interceptors.
- **Policy management and monitoring is integrated into Oracle Enterprise Manager Fusion Middleware Control.** The functions of the Oracle WSM Monitor and the Web Services Manager Control have been integrated into Fusion Middleware Control. This allows you to manage your enterprise from one central location.
- **Oracle WSM Policy Manager enforces additional Web service QoS requirements.** The Oracle WSM Policy Manager manages not only security policies, but it also manages other types of policies such as Message Transmission Optimization Mechanism (MTOM), Reliable Messaging, Addressing, and Management.
- **The Oracle WSM Database is replaced by the Oracle WSM Repository which stores Oracle WSM metadata such as policies, policy sets, assertions templates, and policy usage data.** The Oracle WSM Repository is available as a database (for production use) or as files in the file system (for development use in JDeveloper).

- **Oracle WSM 10g policies have been replaced by Oracle WSM 11g policies.** For a discussion of the differences between the policies in 10g and 11g, see "Comparing Oracle WSM 10g and Oracle WSM 11g Policies" on page 4-3.

Some Oracle WSM 10g features will not be supported in the first release of Oracle Fusion Middleware:

- A subset of Oracle WSM 10*g* components will not be supported in this first release of Oracle Fusion Middleware 11*g*.

  You can continue to use the Oracle WSM 10*g* Gateway components with Oracle WSM 10*g* policies in your applications. For information about Oracle WSM 10*g* interoperability, see "Interoperability with Oracle WSM 10g Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.

- Oracle WSM 10*g* supported policy enforcement agents for third-party application servers, such as IBM WebSphere. Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9) only supports Oracle WebLogic Server. Support for third-party application servers will follow this release.

The comparison between 10*g* and 11*g* components is summarized in Table 4–1 and the components are identified in Figure 4–1 and Figure 4–2.

Table 4–1 Comparison of Oracle WSM 10g and Oracle Fusion Middleware 11g Release 1 (11.1.1.9)

	Description of Functionality	Oracle WSM 10*g* Component	Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9) Component
1	Policy enforcement point	Oracle WSM Server and Client Agents, Oracle WSM Gateway	Oracle WSM Agent which manages the policy interceptors There is no equivalent component for the Oracle WSM Gateway in Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9).
2	GUI Component to author policies and attach policies to Web services	Web Services Manager Control	Oracle Enterprise Manager Fusion Middleware Control
3	Component to manage policies	Oracle WSM Policy Manager	Oracle WSM Policy Manager
4	Component used to monitor Web services data	Oracle WSM Monitor	Oracle Enterprise Manager Fusion Middleware Control and Oracle Enterprise Manager Grid Control
5	Policy Store	Oracle WSM Database	Oracle WSM Repository

Figure 4–1 illustrate the Oracle WSM 10*g* components, and the numbers in Table 4–1 identify the components in this figure.

Figure 4–1 Oracle WSM 10g Components

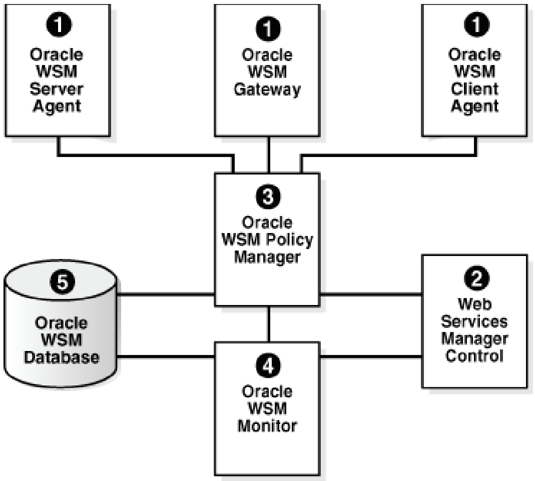
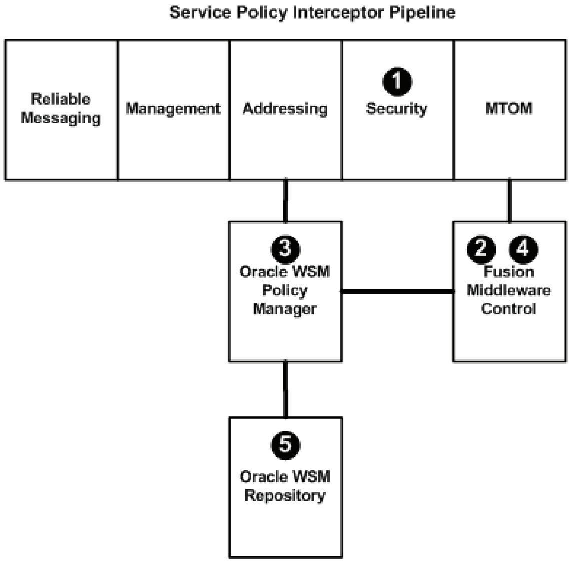


Figure 4–2 shows the Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9) components, and the numbers in Table 4–1 correspond to the components in the figure.

Figure 4–2 Oracle Fusion Middleware 11g Web Services Security Components

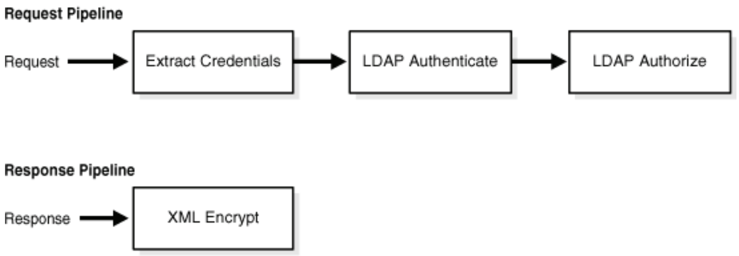

Service Policy Interceptor Pipeline
Reliable Messaging
Management
Addressing
1
Security
MTOM
3
Oracle WSM Policy Manager
2 4
Fusion Middleware Control
5
Oracle WSM Repository

4.2 Comparing Oracle WSM 10*g* and Oracle WSM 11*g* Policies

In both Oracle WSM 10*g* and Oracle WSM 11*g*, policies are used to enforce security. However, the structure of the policies is somewhat different. In Oracle WSM 10*g* a policy consists of a Request Pipeline and a Response Pipeline, each comprised of one or more *policy steps*.

For example, in Figure 4–3, the Request Pipeline consists of the following policy steps: Extract Credentials, LDAP Authenticate, and LDAP Authorize. The Response Pipeline contains a different policy step, XML Encrypt. The Request Pipeline and Response Pipelines can be comprised of different policy steps, and, therefore, different behaviors can be executed in the request and response messages.

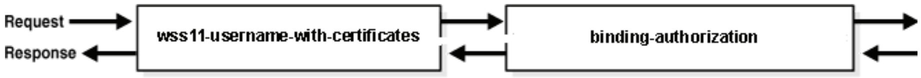
Figure 4–3 Oracle WSM 10g Policy Pipeline



In Oracle WSM 11*g*, policies are comprised of one or more *assertions*, and you control the assertions that are used in the request and response messages. For example, in Figure 4–4, the example 11*g* policy contains two assertions:

1. wss11-username-with-certificates
2. binding-authorization

Figure 4–4 Oracle WSM 11g Policy Pipeline



When the request message is sent to the Web service, the assertions are executed in the order shown. When the response message is returned to the client, the same assertions are executed, but this time in reverse order. The behavior of the assertion for the request message differs from the behavior for the response message. And, in some instances, it is possible that nothing happens on the response. For example, in the example above, the authorization assertion is only executed as part of the request.

For information about how the Oracle WSM 10.1.3 policy steps can be mapped to Oracle WSM 11*g* predefined policies, see "Upgrading Oracle Web Services Manager Policies" in *Oracle Fusion Middleware Upgrade Guide for Oracle SOA Suite, WebCenter Portal, and ADF Release 11g*.

4.3 Comparing Oracle Application Server 10g WS-Security with Oracle WSM 11*g*

The following list identifies the primary enhancements to Oracle WSM 11*g* over Oracle Application Server 10*g* WS-Security:

- **Centralized policy management.** Using the Oracle WSM Policy Manager, you centrally define security and management policies.
- **Custom policy support.** You can create custom policies that support your security and management policy requirements, if the predefined policies do not meet your needs.
- **Toolset used to manage and attach policies.** Security administrators can use Oracle Enterprise Manager Fusion Middleware Control to manage and attach Web services. Developers can attach security policies at development time, using Oracle JDeveloper or other IDE.
- **Policies managed at the enterprise level.** Policies are defined at the enterprise level and not at the application level.

4.4 Interoperability and Upgrade

Oracle WSM 11*g* can interoperate with the following 10.1.3 components:

- Oracle WSM, as described in "Interoperability with Oracle WSM 10g Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.
- Oracle WSM gateways, as described in "Interoperability with Oracle WSM 10g Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.
- Application Server, as described in "Interoperability with Oracle Containers for Java EE (OC4J) 10g Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.

In addition, you can interoperate with the following components:

- WebLogic Web services, as described "Interoperability with Oracle WebLogic Server 11g Web Service Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.
- Microsoft .NET, as described in "Interoperability with Microsoft WCF/.NET 3.5 Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.
- Oracle Service Bus, as described in "Interoperability with Oracle Service Bus 10g Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.
- Axis 1.4 and WSS4J 1.58, as described in "Interoperability with Axis 1.4 and WSS4J 1.58 Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.

You can upgrade the following 10.1.3 features to Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9):

- OC4J Web services 10.1.3 to WebLogic Web services. See "Upgrading Your Java EE Applications" in *Oracle Fusion Middleware Upgrade Guide for Java EE Release 11g*.
- Oracle WSM 10.1.3 policies to Oracle WSM 11g. See "Upgrading Oracle Web Services Manager (WSM) Policies" in *Oracle Fusion Middleware Upgrade Guide for Oracle SOA Suite, WebCenter Portal, and ADF Release 11g*.
- Oracle Containers for Java (OC4J) 10.1.3 security environments to OWSM 11g. See "Upgrading Oracle Containers for Java EE (OC4J) Security Environments" in *Oracle Fusion Middleware Upgrade Guide for Oracle SOA Suite, WebCenter Portal, and ADF Release 11g*.

Part II

Basic Administration

Part II contains the following chapters:

5 Deploying Web Services Applications

This chapter contains the following sections:

- Overview
- Deploying Web Services Applications
- Undeploying a Web Services Application
- Redeploying a Web Services Application

5.1 Overview

As you work with Web services, you will find that you can deploy and undeploy their associated applications in different ways. Follow these guidelines when deploying applications associated with Web services:

- Use Oracle Enterprise Manager Fusion Middleware Control to deploy Java EE applications that require Oracle Metadata Services (MDS) or that take advantage of the Oracle Application Development Framework (Oracle ADF).
- If your application is a SOA composite, use the SOA Composite deployment wizard.
- If your application is a WebCenter application, use Oracle Enterprise Manager Fusion Middleware Control.
- If your application is not a SOA composite or it does not require an MDS repository or ADF connections, then you can deploy your application using Fusion Middleware Control or the Oracle WebLogic Server Administration Console.

> **Note:** To deploy WebLogic Web services, use only the Oracle WebLogic Administration Console.

This chapter provides an overview of the basic procedure for deploying a Web service application. For more information about deploying applications, see "Deploying Applications" in *Oracle Fusion Middleware Administrator's Guide*. In particular, take note of the following sections:

- *Deploying, Undeploying, and Redeploying Java EE Applications*
- *Deploying, Undeploying, and Redeploying Oracle ADF Applications*
- *Deploying, Undeploying, and Redeploying SOA Composite Applications*
- *Deploying, Undeploying, and Redeploying WebCenter Applications*

5.2 Deploying Web Services Applications

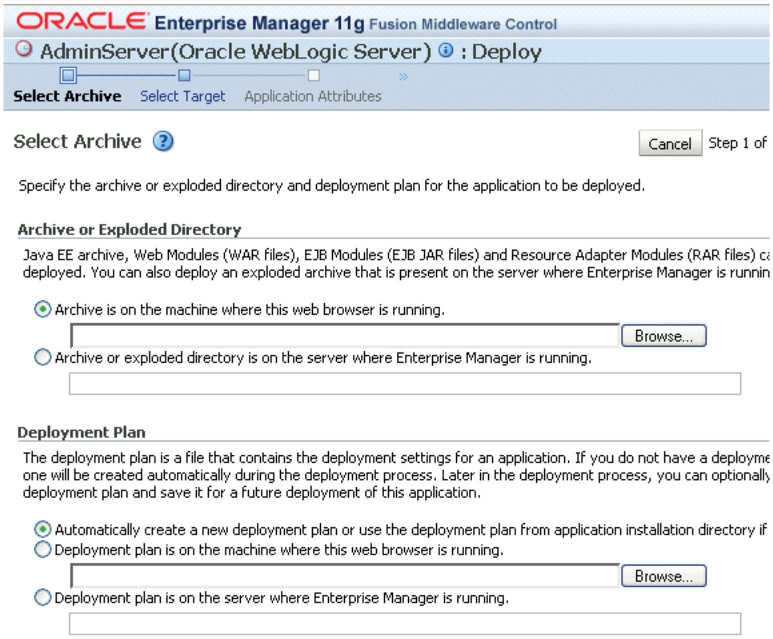
The following is an overview of the basic procedure for deploying a Web service application using the Oracle Enterprise Manager Fusion Middleware Control.

To deploy a Web services application

1. From the navigation pane, expand **WebLogic Domain**.
2. Expand the domain in which you want to deploy the Web service, and then select the instance of the server on which you want to deploy it.
3. Using Fusion Middleware Control, click **WebLogic Server**.
4. Select **Application Deployment**, and then select **Deploy**.

   The first screen of the Deploy process is displayed, as shown in Figure 5–1.

Figure 5–1 Select Archive Page


ORACLE Enterprise Manager 11g Fusion Middleware Control
AdminServer(Oracle WebLogic Server) : Deploy
Select Archive
Select Target
Application Attributes
Cancel
Step 1 of
Browse...
Browse...

Select Archive

Specify the archive or exploded directory and deployment plan for the application to be deployed.

Archive or Exploded Directory

Java EE archive, Web Modules (WAR files), EJB Modules (EJB JAR files) and Resource Adapter Modules (RAR files) ca deployed. You can also deploy an exploded archive that is present on the server where Enterprise Manager is runnin

● Archive is on the machine where this web browser is running.

○ Archive or exploded directory is on the server where Enterprise Manager is running.

Deployment Plan

The deployment plan is a file that contains the deployment settings for an application. If you do not have a deployme one will be created automatically during the deployment process. Later in the deployment process, you can optionally deployment plan and save it for a future deployment of this application.

● Automatically create a new deployment plan or use the deployment plan from application installation directory if

○ Deployment plan is on the machine where this web browser is running.

○ Deployment plan is on the server where Enterprise Manager is running.

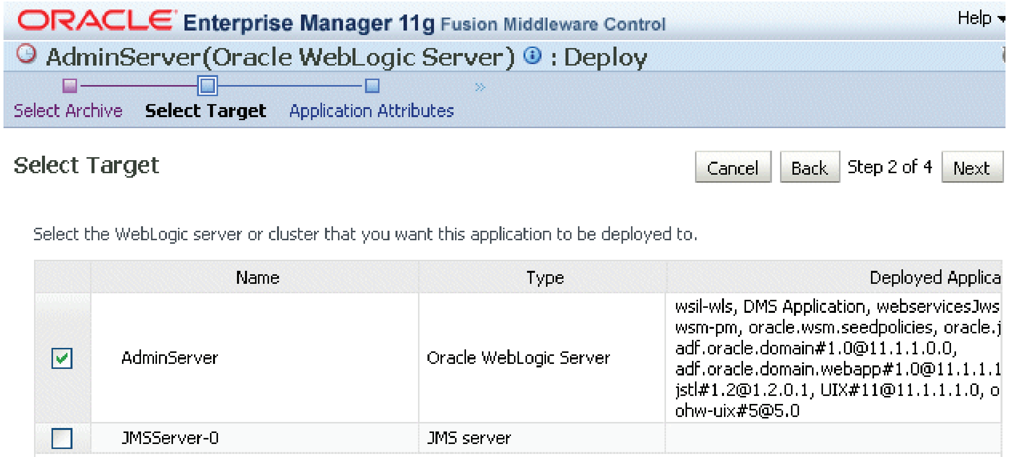
5. Click on one of the following Archive or Exploded Directory options:
   - Archive is on the machine where this web browser is running.
   - Archive or exploded directory is on the server where Enterprise Manager is running.
6. A deployment plan is an XML file that you use to configure an application for deployment to a specific environment. If you do not already have a deployment plan for the Web services application you are deploying, one is created for you when you deploy the application.

   Click one of the following Deployment Plan options:
   - Automatically create a new deployment plan
   - Deployment plan is present on local host
   - Deployment plan is already present on the server where Enterprise Manager is running

7. Click **Next**.

8. On the Select Target page, select the target (WebLogic server or cluster) to which you want this application deployed, and click **Next**.

Figure 5–2 Select Target Page


ORACLE Enterprise Manager 11g Fusion Middleware Control
Help
AdminServer(Oracle WebLogic Server) : Deploy
Select Archive
Select Target
Application Attributes
Cancel
Back
Step 2 of 4
Next

Select Target

Select the WebLogic server or cluster that you want this application to be deployed to.

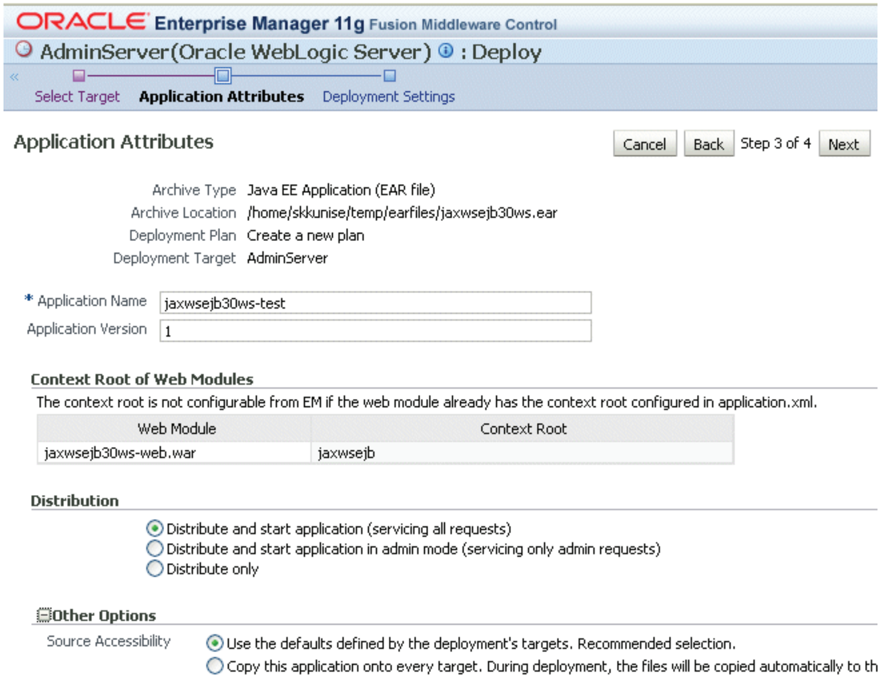
	Name	Type	Deployed Applica
☑	AdminServer	Oracle WebLogic Server	wsil-wls, DMS Application, webservicesJws wsm-pm, oracle.wsm.seedpolicies, oracle.j adf.oracle.domain#1.0@11.1.1.0.0, adf.oracle.domain.webapp#1.0@11.1.1.1 jstl#1.2@1.2.0.1, UIX#11@11.1.1.1.0, o ohw-uix#5@5.0
□	JMSServer-0	JMS server	

9. On the Application Attributes page, enter the attributes for this Web services application, and click **Next.** Application Name is the only required attribute.

   However, if you want to be able to later redeploy this Web service application without first having to undeploy it, you must also assign a version number.

   The context root is the URI for the web module. Each web module or EJB module that contains web services may have a context root.

Figure 5–3 Application Attributes Page


ORACLE Enterprise Manager 11g Fusion Middleware Control
AdminServer(Oracle WebLogic Server) : Deploy
Select Target
Application Attributes
Deployment Settings
Cancel
Back
Step 3 of 4
Next

Application Attributes

Archive Type Java EE Application (EAR file)
Archive Location /home/skkunise/temp/earfiles/jaxwsejb30ws.ear
Deployment Plan Create a new plan
Deployment Target AdminServer

* Application Name jaxwsejb30ws-test
Application Version 1

Context Root of Web Modules

The context root is not configurable from EM if the web module already has the context root configured in application.xml.

Web Module	Context Root
jaxwsejb30ws-web.war	jaxwsejb

Distribution

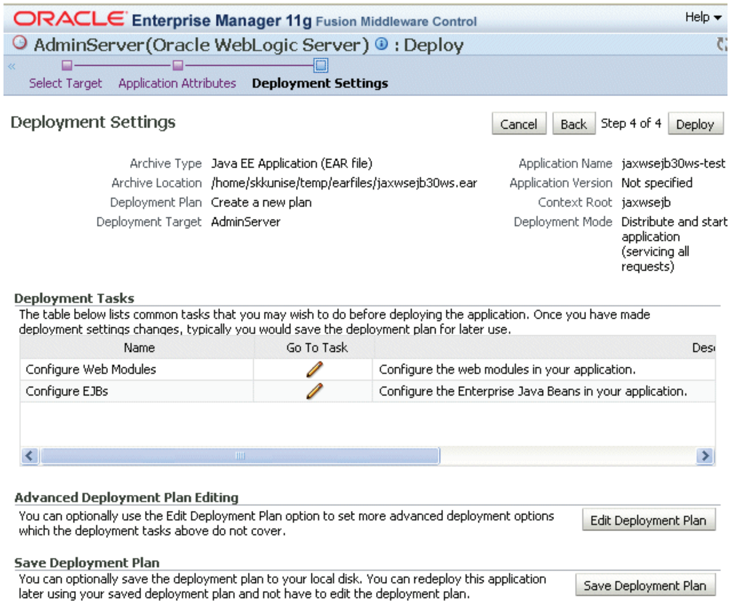
● Distribute and start application (servicing all requests)
○ Distribute and start application in admin mode (servicing only admin requests)
○ Distribute only

Other Options

Source Accessibility
● Use the defaults defined by the deployment's targets. Recommended selection.
○ Copy this application onto every target. During deployment, the files will be copied automatically to th

10. On the Deployment Settings page, edit the deployment settings for this Web services application, as shown in Figure 5–4.

Figure 5–4 Deployment Settings Page



Deployment Settings

Archive Type	Java EE Application (EAR file)	Application Name	jaxwsejb30ws-test
Archive Location	/home/skkunise/temp/earfiles/jaxwsejb30ws.ear	Application Version	Not specified
Deployment Plan	Create a new plan	Context Root	jaxwsejb
Deployment Target	AdminServer	Deployment Mode	Distribute and start application (servicing all requests)

Deployment Tasks

The table below lists common tasks that you may wish to do before deploying the application. Once you have made deployment settings changes, typically you would save the deployment plan for later use.

Name	Go To Task	Des
Configure Web Modules	✎	Configure the web modules in your application.
Configure EJBs	✎	Configure the Enterprise Java Beans in your application.

Advanced Deployment Plan Editing

You can optionally use the Edit Deployment Plan option to set more advanced deployment options which the deployment tasks above do not cover.

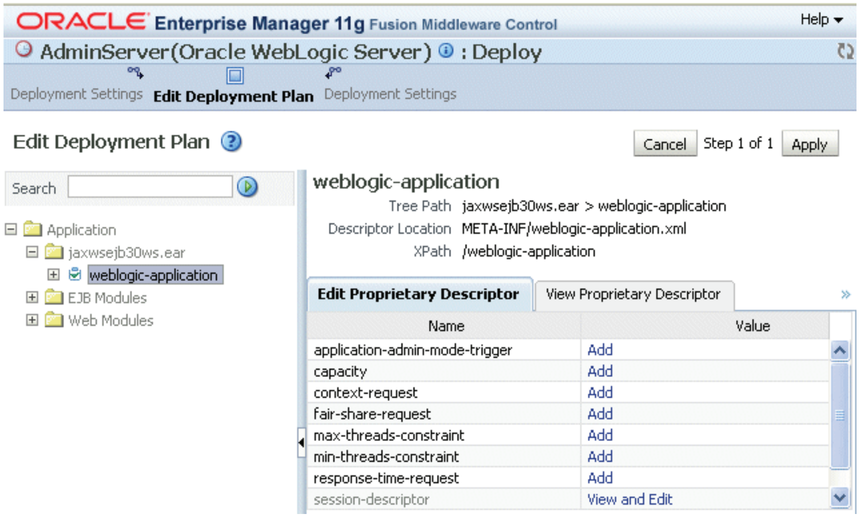
Save Deployment Plan

You can optionally save the deployment plan to your local disk. You can redeploy this application later using your saved deployment plan and not have to edit the deployment plan.

11. To save a copy of the deployment plan to your local system, click **Save Deployment Plan**.

12. To edit the deployment plan, possibly to add advanced deployment options, click **Edit Deployment Plan**. If you do so, the Edit Deployment Plan screen is displayed, as shown in Figure 5–5. After making changes to the deployment plan, click **Apply** to make the change effective.

Figure 5–5 Edit Deployment Plan



weblogic-application

Tree Path	jaxwsejb30ws.ear > weblogic-application
Descriptor Location	META-INF/weblogic-application.xml
XPath	/weblogic-application

Name	Value
application-admin-mode-trigger	Add
capacity	Add
context-request	Add
fair-share-request	Add
max-threads-constraint	Add
min-threads-constraint	Add
response-time-request	Add
session-descriptor	View and Edit

13. Click **Deploy** on the Deployment Settings page. If successful, the Deployment Succeeded screen is displayed.

5.3 Undeploying a Web Services Application

The procedure for undeploying or redeploying a Web service is the same as the procedure for any application.

To undeploy a Web services application

1. From the navigation pane, expand **Application Deployments**, then select the application that you want to undeploy.

   The Application Deployment is displayed

2. Using Fusion Middleware Control, click **Application Deployment**.

3. From the **Application Deployment** menu, select **Application Deployment**, then **Undeploy**.

   The undeploy confirmation page is displayed.

4. Click **Undeploy**.

   Processing messages are displayed.

5. When the operation completes, click **Close**.

5.4 Redeploying a Web Services Application

When you redeploy a Web service application, the running application is automatically stopped and then restarted.

Redeploy an application if:

- You have made changes to the application and you want to make the changes available.
- You have made changes to the deployment plan.
- You want to redeploy an entirely new archive file in a new location.

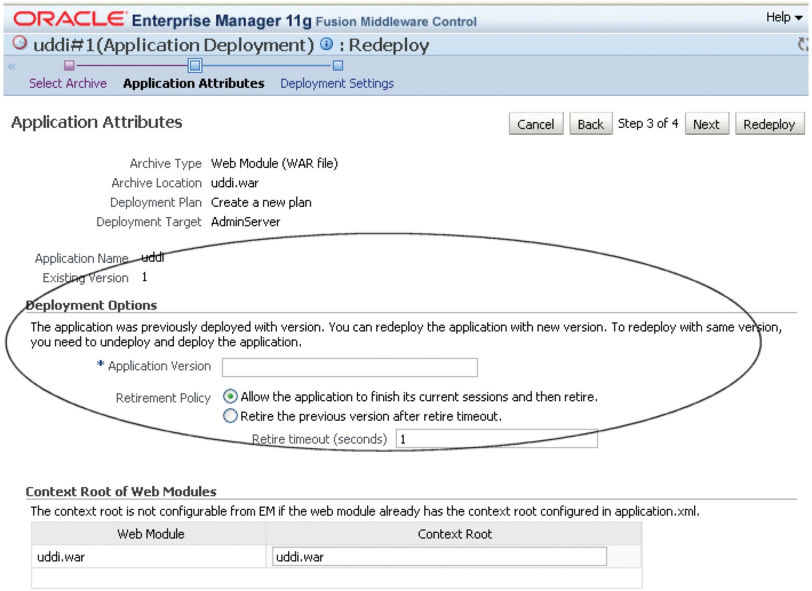
When you redeploy an application, you can redeploy the original archive file or exploded directory, or you can specify a new archive file in place of the original one. You can also change the deployment plan that is associated with the application.

Note: Applications that were previously deployed without a version cannot be redeployed. To redeploy the not-versioned applications, you need to undeploy and deploy the application.

To redeploy a Web services application

The steps that you follow to redeploy a Web service application are identical to those required when you first deployed the application (see Deploying Web Services Applications), with two exceptions: you must redeploy the application with a new version, and you can optionally set the retirement policy for the current version. Both of these actions occur at Step 3 of redeployment process, as shown in Figure 5–6.

Figure 5–6 Setting Application Attributes During Redeploy



Archive Type Web Module (WAR file)
Archive Location uddi.war
Deployment Plan Create a new plan
Deployment Target AdminServer

Application Name uddi
Existing Version 1

Deployment Options

The application was previously deployed with version. You can redeploy the application with new version. To redeploy with same version, you need to undeploy and deploy the application.

* Application Version

Retirement Policy ● Allow the application to finish its current sessions and then retire.
○ Retire the previous version after retire timeout.
Retire timeout (seconds) 1

Context Root of Web Modules

The context root is not configurable from EM if the web module already has the context root configured in application.xml.

Web Module	Context Root
uddi.war	uddi.war

6

Administering Web Services

Oracle Enterprise Manager Fusion Middleware Control is the primary interface that you can use to manage Oracle Fusion Middleware Web Services. You can also use WebLogic Scripting Tool (WLST) commands to perform some configuration tasks for SOA, ADF, and WebCenter services. This chapter describes how to navigate to the pages in Fusion Middleware Control where you perform many of the tasks to manage your Web services, and it describes how to perform basic administration tasks. When applicable, it describes how to perform the task using WLST also. This chapter includes the following sections:

- Viewing All Current Web Services for a Server
- Viewing the Web Services in a Domain Using WLST
- Navigating to the Web Services Summary Page for an Application
- Viewing the Web Services in Your Application
- Viewing the Web Services and References in a SOA Composite
- Viewing the Details for a Web Service Endpoint
- Viewing Web Service Clients
- Displaying the Web Service WSDL Document
- Configuring the Web Service Endpoint
- Enabling or Disabling a Web Service
- Enabling or Disabling RESTful Web Services
- Enabling or Disabling the Display of the Web Service WSDL Document
- Enabling or Disabling the Exchange of Metadata
- Enabling or Disabling the Web Service Test Endpoint
- Validating the Request Message
- Configuring Web Services Atomic Transactions
- Setting the Size of the Request Message
- Configuring Asynchronous Web Services
- Enabling and Disabling MTOM
- Configuring the Web Service Client

6.1 Viewing All Current Web Services for a Server

Follow the procedures below to view all of the currently-deployed Web services for a given server.

To view all current currently-deployed Web services for a given server:

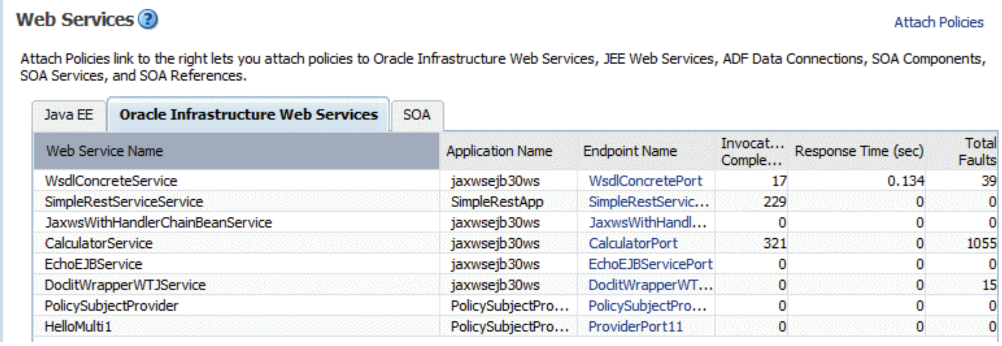
1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to see the Web services.
2. Expand the domain.
3. Select the server for which you want to view all current Web services.
4. Using Fusion Middleware Control, click **WebLogic Server** and then **Web Services**. The server-specific Web Services Summary page appears, as shown in Figure 6–1.

   You can view tabs for Java EE Web services, non-SOA Oracle Web services such as those for ADF and WebCenter, and SOA Web services.

   The tabs that are displayed depend on the Web services deployed on that server.

   From this page you can click **Attach Policies** to attach one or more policies to one or more Web services. Note that attaching policies from this page (bulk attachment) does not perform validation on the policies that you attach.

Figure 6–1 Server-Specific Web Services Summary Page



Attach Policies link to the right lets you attach policies to Oracle Infrastructure Web Services, JEE Web Services, ADF Data Connections, SOA Components, SOA Services, and SOA References.

Web Service Name	Application Name	Endpoint Name	Invocat... Comple...	Response Time (sec)	Total Faults
WsdlConcreteService	jaxwsejb30ws	WsdlConcretePort	17	0.134	39
SimpleRestServiceService	SimpleRestApp	SimpleRestServic...	229	0	0
JaxwsWithHandlerChainBeanService	jaxwsejb30ws	JaxwsWithHandl...	0	0	0
CalculatorService	jaxwsejb30ws	CalculatorPort	321	0	1055
EchoEJBService	jaxwsejb30ws	EchoEJBServicePort	0	0	0
DocitWrapperWTJService	jaxwsejb30ws	DocitWrapperWT...	0	0	15
PolicySubjectProvider	PolicySubjectPro...	PolicySubjectPro...	0	0	0
HelloMulti1	PolicySubjectPro...	ProviderPort11	0	0	0

6.2 Viewing the Web Services in a Domain Using WLST

To view all the current Web services in a domain:

1. Connect to the running instance of WebLogic Server for which you want to view the Web services as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Use the `listWebServices()` WLST command to display a list of the Web services. If you don't specify a Web service application or a SOA composite, the command lists all services in all applications and composites for every server instance in the domain.

   ```
   listWebServices (application,composite,[detail])
   ```

   For example:

   ```
   wls:/jrfServer_domain/serverConfig> listWebServices()

   /jrfServer_domain/jrfServer/jaxws-sut-no-policy :
           moduleName=jaxws-service, moduleType=web,
   serviceName={http://namespace/}TestService
   ```

```
/jrfServer_domain/jrfServer/jaxws-sut :
        moduleName=jaxws-sut-service, moduleType=web,
serviceName={http://namespace/}TestService
```

3. Set the `detail` argument of the `listWebServices` command to `true` to view the endpoint (port) and policy details for all applications and composites in the domain, the secure status of the endpoints, any configuration overrides and constraints, and if the endpoints have a valid configuration. Because you can specify the priority of a global or directly attached policy (using the `reference.priority` configuration override), the `effective` field indicates if directly attached policies are in effect for the endpoint.

   Note: To simplify endpoint management, all directly attached policies are shown in the output regardless of whether they are in effect for the endpoint. In contrast, only globally attached policies that are in effect for the endpoint are displayed.

   An endpoint is considered secure if the policies attached to it (either directly or externally) enforce authentication, authorization, or message protection behaviors.

   Note: The `listWebServices` command output does not include details on SOA components, including policy attachments.

   For example:

```
wls:/jrfServer_domain/serverConfig> listWebServices(detail='true')

/jrfServer_domain/jrfServer_admin/jaxws-sut-no-policy :
        moduleName=jaxws-service, moduleType=web,
serviceName={http://namespace/}TestService
        enableTestPage: true
        enableWSDL: true

                TestPort
http://host.example.com:9315/jaxws-service/TestService
                enable: true
                enableREST: false
                enableSOAP: true
                maxRequestSize: -1
                loggingLevel: NULL
                wsat.flowOption: NEVER
                wsat.version: DEFAULT
                Constraint: No Constraint
                        (global) security : oracle/wss_saml_or_username_token_
service_policy, enabled=true

/policysets/global/all-domains-default-web-service-policies : Domain("*")
                                        reference.priority=1
                Constraint: HTTPHeader('VIRTUAL_HOST_TYPE','external')
                        (global) security : oracle/wss10_message_protection_
service_policy, enabled=true
                                /policysets/global/domainExternal : Domain("*")
                Attached policy or policies are valid; endpoint is secure.
```

```
/jrfServer_domain/jrfServer_admin/jaxws-sut :
        moduleName=jaxws-sut-service, moduleType=web,
serviceName={http://namespace/}TestService
        enableTestPage: true
        enableWSDL: true

                TestPort
http://host.example.com:9315/jaxws-sut-service/TestService
                enable: true
                enableREST: false
                enableSOAP: true
                maxRequestSize: -1
                loggingLevel: NULL
                wsat.flowOption: NEVER
                wsat.version: DEFAULT
                management : oracle/log_policy, enabled=true
                security : oracle/wss_username_token_service_policy ,
enabled=true , effective=false
                Constraint: No Constraint
                        (global) security : oracle/wss_saml_or_username_token_
service_policy, enabled=true

/policysets/global/all-domains-default-web-service-policies : Domain("*")
                                        reference.priority=1
                Constraint: HTTPHeader('VIRTUAL_HOST_TYPE','external')
                        (global) security : oracle/wss10_message_protection_
service_policy, enabled=true
                                /policysets/global/domainExternal : Domain("*")
                Attached policy or policies are valid; endpoint is secure.
```

For more information about the `listWebServices` command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.3 Navigating to the Web Services Summary Page for an Application

Follow the procedure below to navigate to the page where you can see the list of Web services for your application.

To navigate to the Web services summary page for an application:

1. From the navigator pane, click the plus sign (+) for the Application Deployments folder to expose the applications in the domain, and select the application.

   The Application Deployment home page is displayed.

2. Using Fusion Middleware Control, click **Application Deployment**, then click **Web Services**.

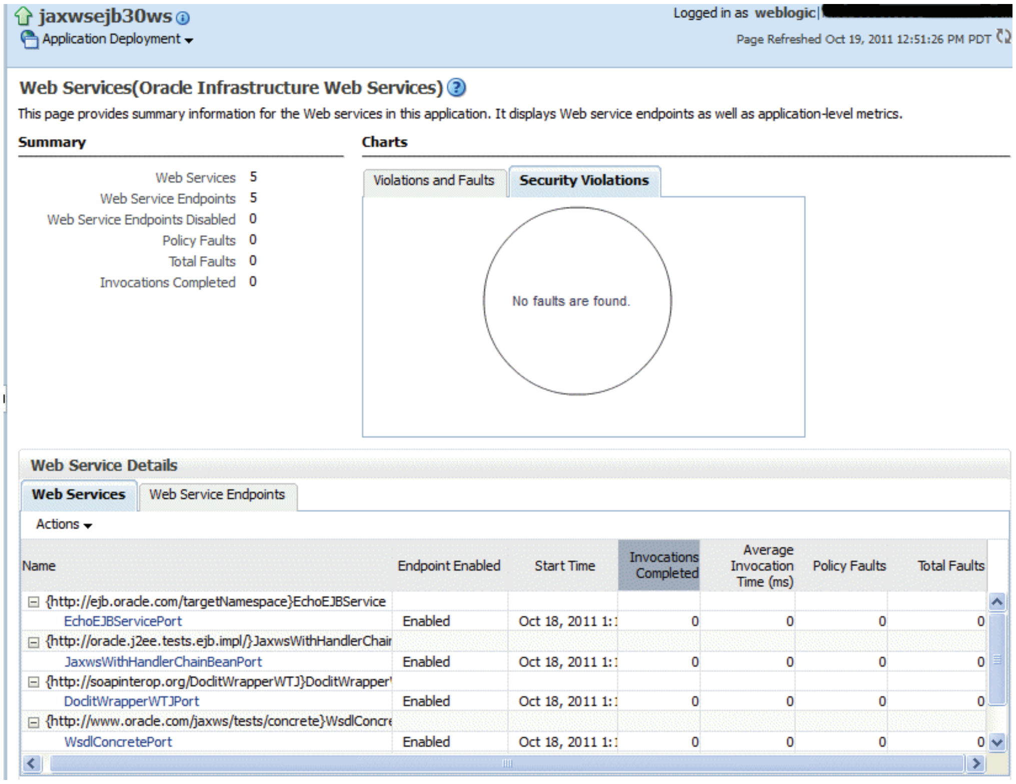
   This takes you to the Web Services summary page for your application. Figure 6–2 shows the Web Services summary page for an ADF or WebCenter application.

   > **Note:** In the Web Service Details section of the page, Oracle Infrastructure Web service provider endpoints display n/a in the Endpoint Enabled column.

   Figure 6–3 shows the Web Services summary page for a WebLogic Java EE application.

Note: The Java EE Web Service Clients tab is displayed only if there are client instances in the application.

Figure 6–2 Web Services Home Page for ADF and WebCenter Applications



jaxwsejb30ws

Application Deployment ▾

Logged in as weblogic| [redacted]

Page Refreshed Oct 19, 2011 12:51:26 PM PDT

Web Services(Oracle Infrastructure Web Services)

This page provides summary information for the Web services in this application. It displays Web service endpoints as well as application-level metrics.

Summary

Web Services	5
Web Service Endpoints	5
Web Service Endpoints Disabled	0
Policy Faults	0
Total Faults	0
Invocations Completed	0

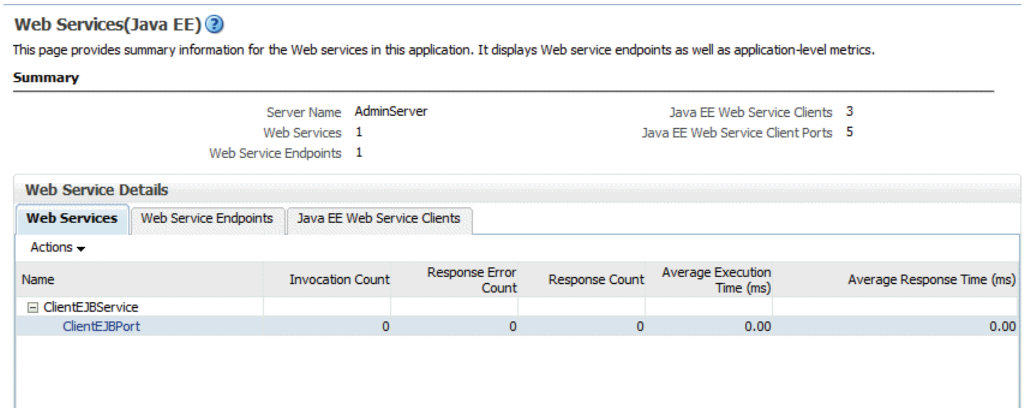
Charts

No faults are found.

Web Service Details

Name	Endpoint Enabled	Start Time	Invocations Completed	Average Invocation Time (ms)	Policy Faults	Total Faults
⊟ {http://ejb.oracle.com/targetNamespace}EchoEJBService						
EchoEJBServicePort	Enabled	Oct 18, 2011 1:	0	0	0	0
⊟ {http://oracle.j2ee.tests.ejb.impl/}JaxwsWithHandlerChair						
JaxwsWithHandlerChainBeanPort	Enabled	Oct 18, 2011 1:	0	0	0	0
⊟ {http://soapinterop.org/DoclitWrapperWTJ}DoclitWrapper'						
DoclitWrapperWTJPort	Enabled	Oct 18, 2011 1:	0	0	0	0
⊟ {http://www.oracle.com/jaxws/tests/concrete}WsdlConcre						
WsdlConcretePort	Enabled	Oct 18, 2011 1:	0	0	0	0

Figure 6–3 Web Services Home Page for WebLogic Java EE Applications



Web Services(Java EE)

This page provides summary information for the Web services in this application. It displays Web service endpoints as well as application-level metrics.

Summary

Server Name	AdminServer	Java EE Web Service Clients	3
Web Services	1	Java EE Web Service Client Ports	5
Web Service Endpoints	1		

Web Service Details

Name	Invocation Count	Response Error Count	Response Count	Average Execution Time (ms)	Average Response Time (ms)
⊟ ClientEJBService					
ClientEJBPort	0	0	0	0.00	0.00

6.4 Viewing the Web Services in Your Application

Use the procedures described in the following sections to view the Web services in your application.

6.4.1 Using Fusion Middleware Control

Navigate to the home page for your Web service, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4. From the Web Services Summary page, you can do the following:

- View the Web services in the application.
- View the Web service configuration, endpoint status, policy faults, and more. (ADF and WebCenter applications only.)
- View and monitor Web services faults, including Security, Reliable Messaging, MTOM, Management, and Service faults. (ADF and WebCenter applications only.)
- View and monitor Security violations, including authentication, authorization, message integrity, and message confidentiality violations. (ADF and WebCenter applications only.)
- Navigate to pages where you can configure your Web services endpoints, including enabling and disabling the endpoint, and attaching policies to Web services.

6.4.2 Using WLST

To view the Web services in your application:

1. Connect to the running instance of WebLogic Server to which the application is deployed as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Use the `listWebServices` WLST command to display a list of the Web services in your application. You must specify the complete application path name to identify the application and the server instance to which it is deployed.

   ```
   listWebServices (application,composite,[detail]
   ```

   For example:

   ```
   wls:/wls-domain/serverConfig>listWebServices("wls-domain/AdminServer/jaxwsejb30
   ws")
   /wls-domain/AdminServer/jaxwsejb30ws:

   moduleName=jaxwsejb,moduleType=web,serviceName={http://namespace/}JaxwsWithHand
   lerChainBeanService
    moduleName=jaxwsejb, moduleType=web,
   serviceName={http://namespace/}WsdlConcreteService
    moduleName=jaxwsejb, moduleType=web,
   serviceName={http://namespace/}EchoEJBService
    moduleName=jaxwsejb, moduleType=web,
   serviceName={http://namespace/}CalculatorService
    moduleName=jaxwsejb, moduleType=web,
   serviceName={http://namespace/}DoclitWrapperWTJService
   ```

For details about the `listWebServices` command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.5 Viewing the Web Services and References in a SOA Composite

Use the following procedure to view the Web services, references, and components in a SOA composite application:

1. From the navigator, click the plus sign (+) for SOA deployments.

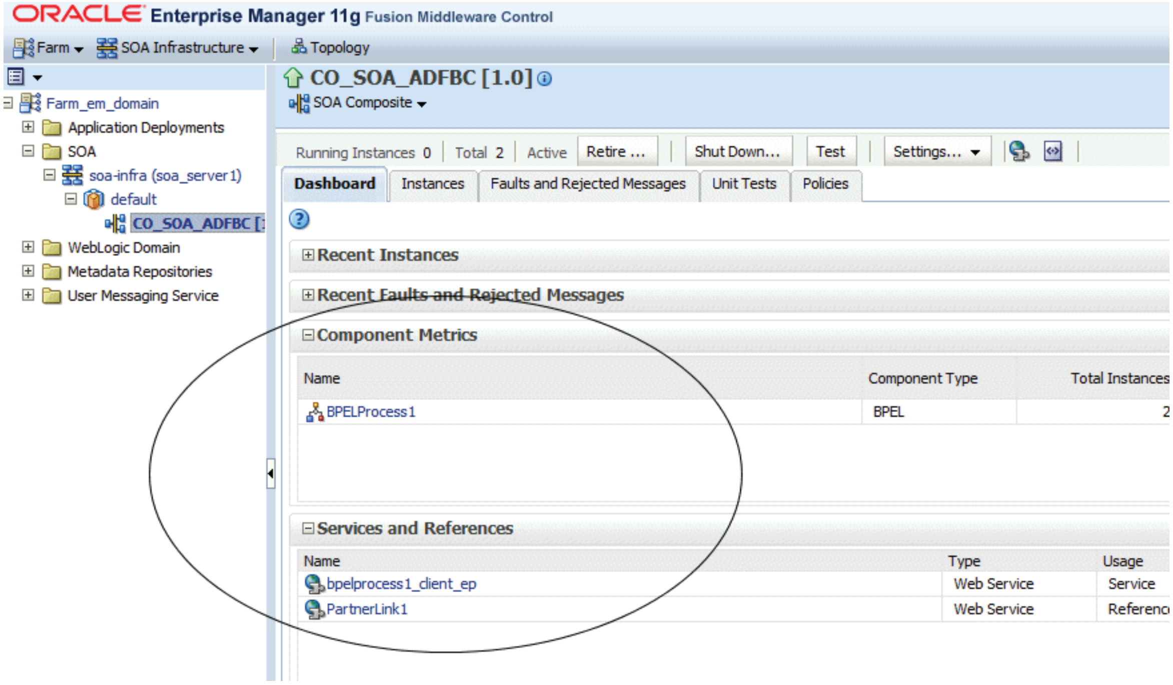
2. Select **soa-infra**, expand the SOA partition (for example, the default partition) and select the target SOA composite application.

   The SOA composite home page displays.

3. Select the **Dashboard** tab if it is not already selected.

   The Component Metrics section of this tab lists the SOA components being used in the composite application, and the Services and References section displays the Web service and reference bindings, as shown in Figure 6–4.

Figure 6–4 SOA Composite Application Dashboard Page



Name	Component Type	Total Instances
BPELProcess1	BPEL	2

Name	Type	Usage
bpelprocess1_client_ep	Web Service	Service
PartnerLink1	Web Service	Referenc

6.6 Viewing the Details for a Web Service Endpoint

Use the procedures described in the following sections to view the details for a Web service endpoint (port) using Oracle Enterprise Manager Fusion Middleware Control and WLST.

6.6.1 Using Fusion Middleware Control

In Fusion Middleware Control, the steps you follow to view the details for a Web service endpoint depend on the application type, as described in the following sections.

To view the details for a non-SOA Oracle Infrastructure or WebLogic Web service endpoint:

1. Navigate to the Web Services Summary page as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.

2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service endpoints if they are not already displayed.

3. Click the name of the endpoint to navigate to the Web Service Endpoint page.

4. From the Web Service Endpoint page, you can do the following:
   - Click the **Operations** tab to see the list of operations for this endpoint.
   - Click the **OWSM Policies** tab to see the policies attached to this endpoint, if the endpoint has a valid configuration, and if it is secure.
   - Click the **Charts** tab to see a graphical display of the faults for this endpoint. (Oracle Infrastructure Web Services only.)
   - Click the **Configuration** tab to see the configuration for this endpoint. (Oracle Infrastructure Web Services only.)

   > **Note:** You can also view details about security violations for an endpoint. For more information, see "Viewing the Security Violations for a Web Service" on page 13-7.

As an alternative method of viewing the details for a Web service endpoint, you can instead navigate to the server-wide Web Services Summary page, as described in "Viewing All Current Web Services for a Server" on page 6-2, which lists all of the Web services, and click the name of the endpoint to navigate to the specific Web Service Endpoint page.

To view the Web service endpoint configuration for a SOA composite application:

1. Navigate to the home page for the SOA composite as described in "Viewing the Web Services and References in a SOA Composite" on page 6-6.
2. In the Services and References section of the page, click the name of the service or reference to display the Service Home or Reference Home page, as appropriate.
3. From the Service Home or Reference Home page, you can do the following:
   - Click the **Dashboard** tab, if it is not already selected, to see a graphic representation of the total incoming messages and faults since server startup, and recently rejected messages, including the message name, time of the fault, and the type of fault (business or system).
   - Click the **Policies** tab to view or change the policies attached to this endpoint.
   - Click the **Faults and Rejected Messages** tab to see a list of faults and rejected messages, including details such as the error message, time of the fault, and the associated composite instance ID.
   - Click the **Properties** tab to view and modify the configuration for this endpoint.

   For additional information about SOA composite endpoints, see "Administering Binding Components" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

6.6.2 Using WLST

To view the details for a Web service endpoint (port):

1. Connect to the running instance of WebLogic Server to which the application is deployed as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Use the `listWebServices` WLST command to display a list of the Web services in your application as described in "Viewing the Web Services in Your Application" on page 6-5.

3. Use the `listWebServicePorts` command to display the endpoint name and endpoint URL for a Web service.

   ```
   listWebServicePorts(application,moduleOrCompName,moduleType,serviceName)
   ```

   For example, to display the endpoint for the `WsdlConcreteService`:

   ```
   wls:/wls-domain/serverConfig>
   listWebServicePorts("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb",
   "web","{http://namespace/}WsdlConcreteService")

   WsdlConcretePort   http://host.example.com:7001/jaxwsejb/WsdlAbstract
   ```

4. Use the `listWebServiceConfiguration` command to view the configuration details for a Web service endpoint.

   ```
   listWebServiceConfiguration(application,moduleOrCompName,moduleType,serviceName
   ,[subjectName])
   ```

   For example, to view the configuration details for the `WsdlConcretePort`:

   ```
   wls:/wls-domain/serverConfig>
   listWebServiceConfiguration("/wls-domain/AdminServer/jaxwsejb30ws",
   "jaxwsejb","web","{http://namespace/}WsdlConcreteService","WsdlConcretePort")
   enable: true
   enableREST: false
   maxRequestSize: -1
   loggingLevel: NULL
   ```

5. Use the `listWebServicePolicies` command to view the policies that are attached to a Web service endpoint.

   ```
   listWebServicePolicies(application,moduleOrCompName,moduleType,serviceName,subj
   ectName)
   ```

   For example, to view the policies attached to the `WsdlConcretePort` endpoint and any policy override settings:

   ```
   wls:/wls_domain/serverConfig> listWebServicePolicies("/wls_
   domain/AdminServer/jaxwsejb30ws",
   "jaxwsejb","web","{http://namespace/}WsdlConcreteService","WsdlConcretePort")

   WsdlConcretePort :
   addressing : oracle/wsaddr_policy , enabled=true
   management : oracle/log_policy , enabled=true
   security : oracle/wss_username_token_service_policy, enabled=true
   Attached policy or policies are valid; endpoint is secure.
   ```

For more information about these WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.7 Viewing Web Service Clients

The following sections describe how to view Web service clients for your application.

6.7.1 Using Fusion Middleware Control

The steps you follow to view a Web service client depend on the application type (SOA reference, ADF DC, WebCenter, or asynchronous Callback client), as described in the following sections.

6.7.1.1 Viewing SOA References

Use the following procedure to view a SOA reference client:

1. From the navigator pane, click the plus sign (+) for SOA deployments.
2. Select **soa-infra**, expand the SOA partition (for example, the default partition) and select the target SOA composite application.

   The SOA composite home page displays.
3. Click the **Dashboard** tab, if it is not already selected.
4. In the Services and References portion of the page, select the SOA reference to view.
5. In the Reference Home page, click the tabs to view the client data.

6.7.1.2 Viewing Connection-Based Web Service Clients

Use the following procedure to view a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client:

1. From the navigator pane, click the plus sign (+) for the Application Deployments folder to expose the applications in the farm, and select the application.

   The Application Deployment home page is displayed.
2. From the **Application Deployment** menu, select **ADF**, and then **Configure ADF Connections**.
3. On the ADF Connections Configuration page, select a connection from the Web Service Connections section of the page, and then select the endpoint from the **Configure Web Service** list.
4. In the Configure Web Service page, click the tabs to view the client data.

6.7.1.3 Viewing WebCenter Portlets

Use the following procedure to view a WebCenter portlet:

1. From the navigator pane, click the plus sign (+) for the WebCenter folder and WebCenter Spaces folder to display the WebCenter spaces.
2. Click the name of the WebCenter space to view.
3. From the WebCenter menu, select **Settings** and **Service Configuration**.

   The Webcenter Service Configuration page is displayed.
4. Select **Portlet Producers** to view the WebCenter portlets.

6.7.1.4 Viewing Java EE Web Service Clients

Use the following procedure to view Java EE Web service clients:

1. From the navigator pane, click the plus sign (+) for the Application Deployments folder to expose the applications in the farm, and select the Java EE application.

   The Application Deployment home page is displayed.

2. From the **Application Deployment** menu, select **Web Services**.

   The Web Services (Java EE) home page is displayed.

3. Select the **Java EE Web Service Clients** tab to view the clients in the application.

   - Use the **Monitoring** tab to view the run-time client instances in the application.
   - Use the **Configuration** tab to view the client ports and attach or detach policies.

6.7.1.5 Viewing Asynchronous Web Service Callback Clients

Use the following procedure to view an asynchronous Web service Callback client. Callback clients are used only by asynchronous Web services to return the response to the caller. For more information, see "Developing Asynchronous Web Services" in *Developer's Guide for Oracle Infrastructure Web Services*.

1. Navigate to the endpoint for the asynchronous Web service, as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.

2. Click **Callback Client** in the upper right portion of the endpoint page.

6.7.2 Using WLST

Use the following procedure to view the Web service clients using WLST commands:

1. Connect to the running instance of WebLogic Server to which the application is deployed as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Use the listWebServiceClients WLST command to display a list of the Web service clients.

   ```
   listWebServiceClients(application,composite,[detail])
   ```

   This command enables you to list the clients for an application, a SOA composite, or a domain. To list the client information for an application or SOA composite, specify the appropriate argument. If you do not specify an application or SOA composite, the command outputs information, including the module name, module type, and SOA reference name for all the Web service clients in all applications and composites in every server instance in the domain. To view details about each client, including the endpoint and policies, set the detail argument to true.

   For example:

   ```
   wls:/soainfra/serverConfig> listWebServiceClients(detail=true)

   /soainfra/soa_server1/soa-infra :
           compositeName=default/SampleSOAFirstPrj[1.0], moduleType=soa,
   serviceRefName=ReferenceToSecondSOA
                   BPELProcess1_pt   serviceWSDLURI=
                     http://localhost:8001/soa-infra/services/default/
                     SampleSOASecondPrj/BPELProcess1.wsdl
                   oracle.webservices.contentTransferEncoding=base64
                   oracle.webservices.charsetEncoding=UTF-8
                   oracle.webservices.operationStyleProperty=document
                   oracle.webservices.soapVersion=soap1.1
                   oracle.webservices.chunkSize=4096
                   oracle.webservices.preemptiveBasicAuth=false
                   oracle.webservices.session.maintain=false
   ```

```
                oracle.webservices.encodingStyleProperty=
                  http://schemas.xmlsoap.org/soap/encoding/
                oracle.webservices.donotChunk=true
                No attached policies found; endpoint is not secure.


/soainfra/AdminServer/ADFDCApp :
        moduleName=adfdc, moduleType=wsconn, serviceRefName=AppModuleService
                AppModuleServiceSoapHttpPort    serviceWSDLURI=
                   http://localhost:8001/ADF-App-context-root/
                   AppModuleService?wsdl
                security : oracle/wss_username_token_client_policy,
enabled=true
                Attached policy or policies are valid; endpoint is secure.
```

Note that the output displays SOA references (using the `serviceRefName` argument) for the SOA composites `default/SampleSOAFirstPrj[1.0]`. To list the SOA references for a SOA composite, specify the composite name in the command, for example `listWebServiceClients(None,'default/SampleSOAFirstPrj[1.0]')`.

ADF and WebCenter clients are specified by the `moduleType=wsconn` argument in the output.

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

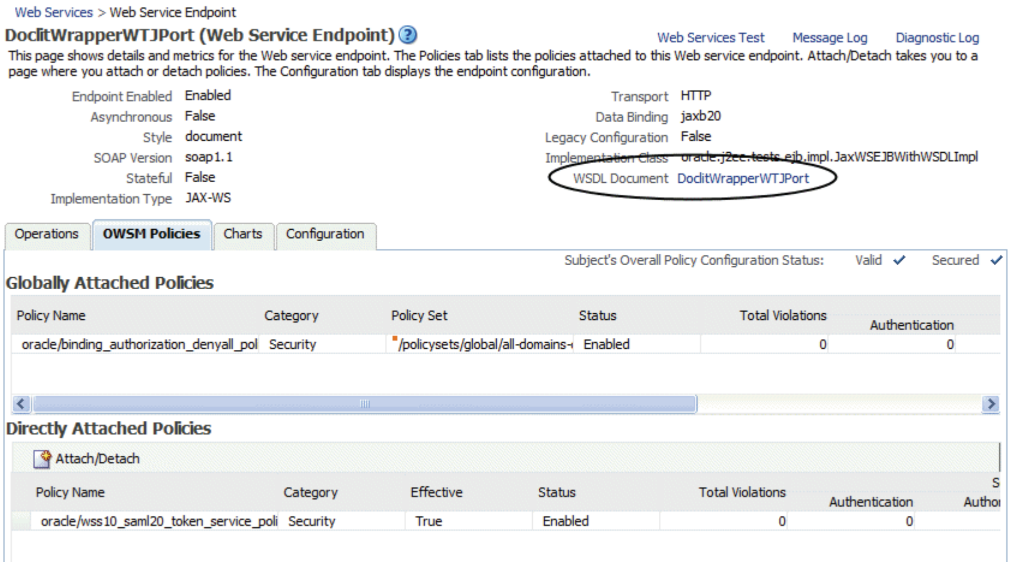
6.8 Displaying the Web Service WSDL Document

Follow the procedure below to display the WSDL document for a Web service.

To display the WSDL document for a Web service:

1. Navigate to the Web Services Summary page.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service endpoints if they are not already displayed.
3. Click the name of the endpoint to navigate to the Web Service Endpoint page.
4. In the WSDL Document field, click the endpoint name to display the WSDL for the Web service (Figure 6–5).

Figure 6–5 Web Service Endpoint page with Web Service WSDL



Web Services > Web Service Endpoint

DoclitWrapperWTJPort (Web Service Endpoint)

This page shows details and metrics for the Web service endpoint. The Policies tab lists the policies attached to this Web service endpoint. Attach/Detach takes you to a page where you attach or detach policies. The Configuration tab displays the endpoint configuration.

Endpoint Enabled	Enabled	Transport	HTTP
Asynchronous	False	Data Binding	jaxb20
Style	document	Legacy Configuration	False
SOAP Version	soap1.1	Implementation Class	oracle.j2ee.tests.ejb.impl.JaxWSEJBWithWSDLImpl
Stateful	False	WSDL Document	DoclitWrapperWTJPort
Implementation Type	JAX-WS		

Subject's Overall Policy Configuration Status: Valid ✓ Secured ✓

Globally Attached Policies

Policy Name	Category	Policy Set	Status	Total Violations	Authentication
oracle/binding_authorization_denyall_pol	Security	/policysets/global/all-domains-	Enabled	0	0

Directly Attached Policies

Policy Name	Category	Effective	Status	Total Violations	Authentication	S Autho
oracle/wss10_saml20_token_service_poli	Security	True	Enabled	0	0	

6.9 Configuring the Web Service Endpoint

Follow the procedures below to configure the Web service endpoint (or port).

> **Note:** The procedures described in this section apply to Oracle Infrastructure Web services and providers only.
>
> Oracle Infrastructure Web service providers implement the java.xml.ws.Provider interface. On the Web Service Endpoint page, they display the Implementation Class and provide a subset of configuration properties.

6.9.1 Using Fusion Middleware Control

Use the following procedure to configure the Web service endpoint using Fusion Middleware Control:

1. Navigate to the Web Service Endpoint page, or the Service Home page (for SOA composites), as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click the **Configuration** tab. For SOA composites, click the **Properties** tab.
3. Set the configuration attributes and click **Apply.**

   For more information about setting the configuration attributes, see:

   - "Enabling or Disabling a Web Service" on page 6-16
   - "Enabling or Disabling RESTful Web Services" on page 6-17
   - "Enabling or Disabling the Display of the Web Service WSDL Document" on page 6-18
   - "Enabling or Disabling the Exchange of Metadata" on page 6-19
   - "Enabling or Disabling the Web Service Test Endpoint" on page 6-20
   - "Setting the Log Level for Diagnostic Logs" on page 16-21

- "Validating the Request Message" on page 6-21
- "Configuring Web Services Atomic Transactions" on page 6-21
- "Setting the Size of the Request Message" on page 6-24
- "Configuring Asynchronous Web Services" on page 6-26

4. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

> **Note:** You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

6.9.2 Using WLST

Use the following procedure to configure the Web service endpoint (port) using WLST:

1. Connect to the running instance of WebLogic Server to which the application is deployed as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Use the `listWebServices` WLST command to display a list of the Web services in your application as described in "Viewing the Web Services in Your Application" on page 6-5.
3. Use the `listWebServicePorts` command to display the endpoint name and endpoint URL for a Web service.

   ```
   listWebServicePorts(application,moduleOrCompName,moduleType,serviceName)
   ```

   For example, to display the endpoint for the `WsdlConcreteService`:

   ```
   wls:/wls-domain/serverConfig>
   listWebServicePorts("/wls-domain/AdminServer/jaxwsejb30ws",None,"web",
   "{http://namespace/}WsdlConcreteService")

   WsdlConcretePort   http://host.example.com:7001/jaxwsejb/WsdlAbstract
   ```

4. Use the `listWebServiceConfiguration` command to view the configuration details for a Web service endpoint.

   ```
   listWebServiceConfiguration(application,moduleOrCompName,moduleType,serviceName
   ,[subjectName])
   ```

   For example, to view the configuration details for the `WsdlConcretePort`:

   ```
   wls:/wls-domain/serverConfig>
   listWebServiceConfiguration("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb",
   "web","{http://namespace/}WsdlConcreteService","WsdlConcretePort")
   enable: true
   enableREST: false
   maxRequestSize: -1
   loggingLevel: NULL
   ```

Alternatively, you can set the `detail` argument to `true` in the `listWebServices` command to view the configuration details for the endpoint as shown in "Viewing the Web Services in a Domain Using WLST" in "Viewing All Current Web Services for a Server" on page 6-2.

5. Use the `setWebServiceConfiguration` command to set or change the endpoint configuration. Specify the properties to be set or changed using the `itemProperties` argument.

```
setWebServiceConfiguration(application,moduleOrCompName,moduleType,
serviceName,subjectName,itemProperties)
```

For example, to change the logging level to SEVERE for the `WsdlConcretePort`, use the following command:

```
wls:/wls-domain/serverConfig>
setWebServiceConfiguration("/wls-domain/AdminServer/jaxwsejb30ws",
"jaxwsejb","web","{http://namespace/}WsdlConcreteService","WsdlConcretePort",
[("loggingLevel","SEVERE")])

Please restart application to uptake the policy changes.
```

For more information about the configurable properties, see:

- "Enabling or Disabling a Web Service" on page 6-16
- "Enabling or Disabling RESTful Web Services" on page 6-17
- "Enabling or Disabling the Display of the Web Service WSDL Document" on page 6-18
- "Enabling or Disabling the Web Service Test Endpoint" on page 6-20
- "Configuring Web Services Atomic Transactions" on page 6-21
- "Setting the Size of the Request Message" on page 6-24
- "Setting the Log Level for Diagnostic Logs" on page 16-21

Note: If any configuration item contains an unrecognized property name or an invalid value, this set command is rejected and an error message is displayed.

6. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

For more information about these WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.10 Enabling or Disabling a Web Service

When a Web service application is deployed, the Web service endpoint is enabled by default if no errors are encountered. If there are errors, the Web service application is deployed, but the Web service endpoint is not enabled.

You may need to temporarily make a Web service unavailable by disabling the Web service. For example, you may need to correct an invalid policy reference. When you disable a Web service, requests to the Web service will fail. To disable a Web service, you must make the endpoint on which the Web service receives requests unavailable.

Note: The procedures described in this section apply to Oracle Infrastructure Web services only.

6.10.1 Using Fusion Middleware Control

To disable an ADF or WebCenter Web service endpoint:

1. Navigate to the Web Services Summary page.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service endpoints if they are not already displayed.
3. Click the name of the endpoint to navigate to the Web Service Endpoint page.
4. From the Web Service Endpoint page, click the **Configuration** tab.
5. In the Endpoint Enabled field, select **Disabled** from the menu, and click **Apply.**
6. Restart the application that uses the Web service.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

6.10.2 Using WLST

To disable a Web service endpoint (port) using WLST, use the `setWebServiceConfiguration` command. Set the `enable` property of the `itemProperties` argument to `false` to disable the endpoint and to `true` to enable it.

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Endpoint" on page 6-13.

For example, to disable the endpoint `WsdlConcretePort`, use the following command:

```
wls:/wls-domain/serverConfig> setWebServiceConfiguration
("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb","web","{http://namespace/}WsdlC
oncreteService",
"WsdlConcretePort",[("enable","false")])

Please restart application to uptake the policy changes.
```

For more information about this WLST command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.11 Enabling or Disabling RESTful Web Services

You can enable or disable a Web services endpoint to accept messages in Representational State Transfer (REST) format.

> **Note:** The procedures described in this section apply to Oracle Infrastructure Web services only.

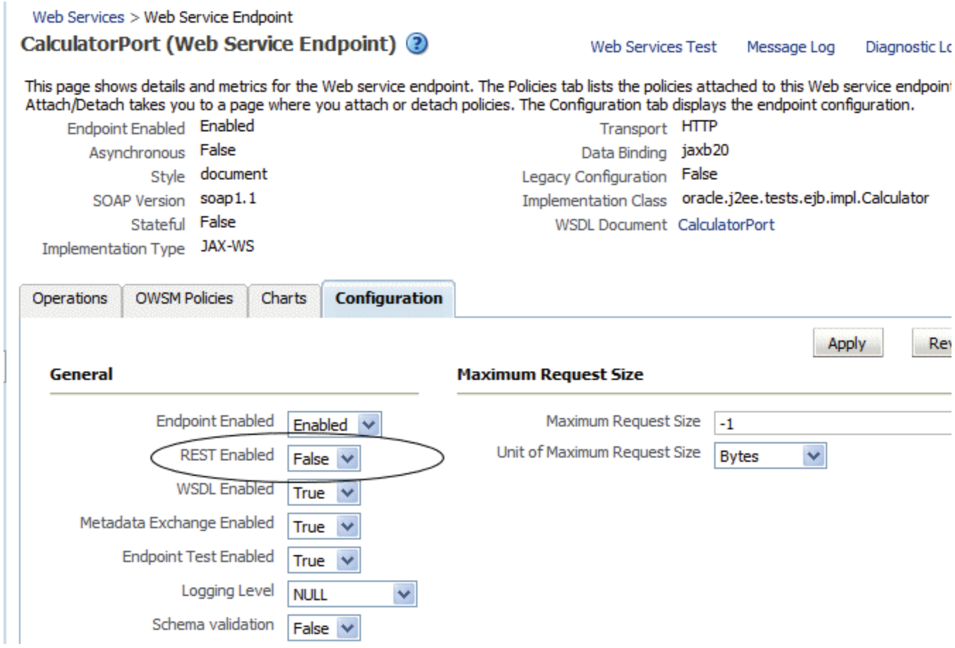
6.11.1 Using Fusion Middleware Control

To enable or disable Web service styles:

1. Navigate to the Web Service Endpoint page, or the Service Home page (for SOA composites), as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click the **Configuration** tab. For SOA composites, click the **Properties** tab.
3. In the REST Enabled field, select **True** from the menu to enable REST, or select **False** to disable REST, and click **Apply**.

   Figure 6–3 indicates the location of the REST Enabled field for an ADF or WebCenter endpoint.

Figure 6–6 Enabling and Disabling RESTful Web Services



CalculatorPort (Web Service Endpoint)

This page shows details and metrics for the Web service endpoint. The Policies tab lists the policies attached to this Web service endpoin Attach/Detach takes you to a page where you attach or detach policies. The Configuration tab displays the endpoint configuration.

Endpoint Enabled	Enabled	Transport	HTTP
Asynchronous	False	Data Binding	jaxb20
Style	document	Legacy Configuration	False
SOAP Version	soap1.1	Implementation Class	oracle.j2ee.tests.ejb.impl.Calculator
Stateful	False	WSDL Document	CalculatorPort
Implementation Type	JAX-WS		

General

Endpoint Enabled	Enabled
REST Enabled	False
WSDL Enabled	True
Metadata Exchange Enabled	True
Endpoint Test Enabled	True
Logging Level	NULL
Schema validation	False

Maximum Request Size

Maximum Request Size	-1
Unit of Maximum Request Size	Bytes

4. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

6.11.2 Using WLST

To enable or disable a Web services endpoint (port) to accept messages in REST format using WLST, use the `setWebServiceConfiguration` command. Set the `enableREST` property of the `itemProperties` argument to `true` to enable REST and to `false` to disable it.

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Endpoint" on page 6-13.

For example, to enable the REST format for the `WsdlConcretePort`, use the following command:

```
wls:/wls-domain/serverConfig> setWebServiceConfiguration
("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb","web","{http://namespace/}WsdlC
oncreteService",
"WsdlConcretePort",[("enableREST","true")])

Please restart application to uptake the policy changes.
```

For more information about this WLST command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.12 Enabling or Disabling the Display of the Web Service WSDL Document

The following procedures describe how to enable or disable the display of the Web service WSDL document.

Note: The procedures described in this section apply to Oracle Infrastructure Web services and providers only.

6.12.1 Using Fusion Middleware Control

To enable or disable the display of the Web service WSDL document:

1. Navigate to the Web Service Endpoint page, or the Service Home page (for SOA composites), as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click the **Configuration** tab. For SOA composites, click the **Properties** tab.
3. From the WSDL Enabled field, select **True** from the menu to enable the display of the WSDL or **False** to disable the display of the WSDL, and click **Apply.**
4. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

> **Note:** You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

6.12.2 Using WLST

To enable or disable the display of a WSDL document for a Web service endpoint (port), use the `setWebServiceConfiguration` command. Set the `enableWSDL` property of the `itemProperties` argument to `true` to enable display the WSDL and to `false` to disable it.

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Endpoint" on page 6-13.

For example, to enable the WSDL display for the `WsdlConcretePort`, use the following command:

```
wls:/wls-domain/serverConfig> setWebServiceConfiguration
("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb","web",
"{http://namespace/}WsdlConcreteService","WsdlConcretePort",[("enableWSDL","true")
])

Please restart application to uptake the policy changes.
```

For more information about this WLST command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.13 Enabling or Disabling the Exchange of Metadata

The following procedure describes how to enable or disable the exchange of Web service metadata.

> **Note:** The procedure described in this section applies to Oracle Infrastructure Web services only.

To enable or disable the exchange of metadata:

1. Navigate to the Web Service Endpoint page, or the Service Home page (for SOA composites), as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click the **Configuration** tab. For SOA composites, click the **Properties** tab.
3. In the Metadata Exchange Enabled field, select **True** from the menu to enable the exchange of metadata or **False** to disable the exchange of metadata, and click **Apply.**
4. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

6.14 Enabling or Disabling the Web Service Test Endpoint

The following procedures describes how to enable or disable the Web service test endpoint using Fusion Middleware Control and WLST.

Note: The procedures described in this section apply to Oracle Infrastructure Web services and providers only.

6.14.1 Using Fusion Middleware Control

To enable or disable the Web service test endpoint:

Note: This flag does not control the availability of the **Web Services Test** link.

1. Navigate to the Web Service Endpoint page, or the Service Home page (for SOA composites), as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click the **Configuration** tab. For SOA composites, click the **Properties** tab.
3. In the Endpoint Test Enabled field, select **True** from the menu to enable the test endpoint or **False** to disable the test endpoint, and click **Apply.**
4. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

6.14.2 Using WLST

To enable or disable the Web service test endpoint, use the `setWebServiceConfiguration` command. Set the `enableTestPage` property of the `itemProperties` argument to `true` to enable the test endpoint and to `false` to disable it.

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Endpoint" on page 6-13.

For example, to enable the test endpoint for the `WsdlConcretePort`, use the following command:

```
wls:/wls-domain/serverConfig> setWebServiceConfiguration
("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb","web","{http://namespace/}WsdlC
oncreteService",
"WsdlConcretePort",[("enableTestPage","true")])

Please restart application to uptake the policy changes.
```

For more information about this WLST command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.15 Validating the Request Message

The following procedure describes how to enable or disable the validation of the request message against the schema.

> **Note:** The procedure described in this section applies to Oracle Infrastructure Web services only.

To enable or disable schema validation:

1. Navigate to the Web Service Endpoint page, or the Service Home page (for SOA composites), as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click the **Configuration** tab. For SOA composites, click the **Properties** tab.
3. In the Schema validation field, select **True** from the menu to enable schema validation or **False** to disable schema validation, and click **Apply.**
4. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

> **Note:** You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

6.16 Configuring Web Services Atomic Transactions

WebLogic Web services support the WS-Coordination and WS-AtomicTransaction (WS-AT) specifications. Therefore, you can configure Web services atomic transactions to enable interoperability between Oracle WebLogic Server and other vendor's transaction processing systems, such as WebSphere, Microsoft .NET, and so on.

Web services atomic transactions are supported for WebLogic JAX-WS Web services and SOA Web services and references. You can enable and configure Web services atomic transactions at design time as described in the following topics:

- "Using Web Services Atomic Transactions" in *Programming Advanced Features of JAX-WS Web Services for Oracle WebLogic Server*
- "WS-Atomic Transaction Support" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*

For WebLogic JAX-WS Web services, you can configure Web services atomic transactions at deployment time using the WebLogic Server Administration Console. For more information, see "Configure Web service atomic transactions" in the *Oracle WebLogic Server Administration Console Help*.

For SOA Web services and references, you can configure Web services atomic transactions at deployment time, on the service or reference endpoint, using Oracle Enterprise Manager Fusion Middleware Control or WLST. Refer to the following sections for detailed procedures using both interfaces.

For information about configuring Web service atomic transactions for SOA references, see "Configuring the Web Service Client" on page 6-27.

6.16.1 Using Fusion Middleware Control

To configure atomic transactions for a SOA Web service:

1. Navigate to the SOA composite home page as described in "Viewing the Web Services and References in a SOA Composite" on page 6-6.
2. In the Services and References section of the page, select the service to be configured.
3. In the Service Home page, click the **Properties** tab.
4. In the **Atomic Transaction Version** field, select the version of the Web service atomic transaction coordination context that is supported for the SOA service. The value specified must be consistent across the entire transaction. Valid values are:
   - **WSAT10**
   - **WSAT11**
   - **WSAT12**
   - **Default**

   If you select **Default**, all three versions are accepted.

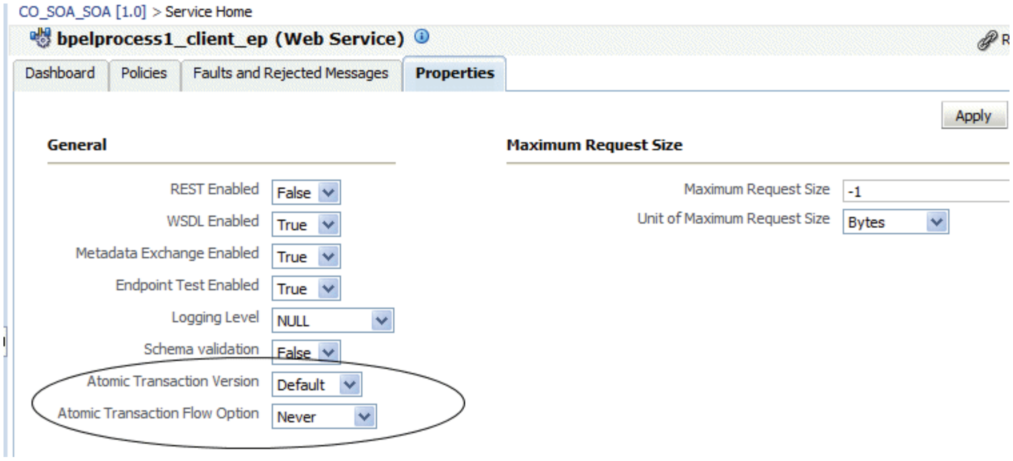
   > **Note:** This property works with SOA Web services that have synchronous-only operations and with Web services that have both synchronous and asynchronous operations. It does not work with SOA Web services with asynchronous-only operations.

5. In the **Atomic Transaction Flow Option** field, select whether the transaction coordination context is to be passed with the transaction flow into the SOA Web service.

   Valid values on the SOA Web service are:
   - **Never** – Do not export transaction coordination context. This is the default.
   - **Supports** – Export transaction coordination context if transaction is available.
   - **Mandatory** – Export transaction coordination context. An exception is thrown if there is no active transaction.

> **Note:** This property works with Web services that have synchronous-only operations or that have combined synchronous and asynchronous operations. It does not work with Web services with asynchronous-only operations.

Figure 6–7 Configuring SOA Web Services Atomic Transactions



CO_SOA_SOA [1.0] > Service Home

bpelprocess1_client_ep (Web Service)

General

REST Enabled: False
WSDL Enabled: True
Metadata Exchange Enabled: True
Endpoint Test Enabled: True
Logging Level: NULL
Schema validation: False
Atomic Transaction Version: Default
Atomic Transaction Flow Option: Never

Maximum Request Size

Maximum Request Size: -1
Unit of Maximum Request Size: Bytes

6. Click **Apply**.

6.16.2 Using WLST

To configure atomic transactions for a SOA Web service endpoint using WLST, use the `setWebServiceConfiguration` command. The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Endpoint" on page 6-13.

> **Note:** To configure Web service atomic transactions for SOA references, you use the `setWebServiceClientStubProperty` command. For additional information, see "Configuring the Web Service Client" on page 6-27.

Specify values for the `itemProperties` argument as described in the following table.

Table 6–1 SOA Web Service Atomic Transaction WLST Configuration Properties

Property	Description	Valid Values
`wsat.flowOption`	Atomic transaction flow option	■ `"NEVER"` – Do not export transaction coordination context. This is the default. ■ `"SUPPORTS"` – Export transaction coordination context if transaction is available. ■ `"MANDATORY"` – Export coordination context. An exception is thrown if there is no active transaction.
`wsat.version`	Atomic transaction version	■ `"WSAT10"` ■ `"WSAT11"` ■ `"WSAT12"` ■ `"DEFAULT"`—If you specify `DEFAULT`, all three versions are accepted.

For example, to configure atomic transactions for the TaskService_pt Web service endpoint of the default/SimpleApproval[1.0] SOA composite application, use the following command:

```
wls:/soa-infra/serverConfig>setWebServiceConfiguration
("soa-infra","default/SimpleApproval[1.0]","soa","client",
"TaskService_pt",[("wsat.flowOption","MANDATORY"),("wsat.version", "DEFAULT")])
```

To verify the settings, use the list(None,None,true) command:

```
wls:/soainfra/serverConfig>listWebServices(None,None,true)
  /soainfra/soa_server1/soa-infra:
        compositeName=default/SimpleApproval[1.0], moduleType=soa,
        serviceName=client
        enableTestPage: true
        enableWSDL: true
           TaskService_pt
http://myhost:8001/soa-infra/services/default/SimpleApproval!1.0/client
                enable: true
                enableREST: false
                enableSOAP: true
                maxRequestSize: -1
                loggingLevel: NULL
                wsat.flowOption: MANDATORY
                wsat.version: DEFAULT
                No policies attached; endpoint is not secure.
```

Note: The listWebServices command output does not include details on SOA components, including policy attachments.

For more information about this WLST command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.17 Setting the Size of the Request Message

The maximum size of the request message to the Web service can be configured using the procedures provided in the following sections.

Note: The procedures described in this section apply to Oracle Infrastructure Web services and providers only.

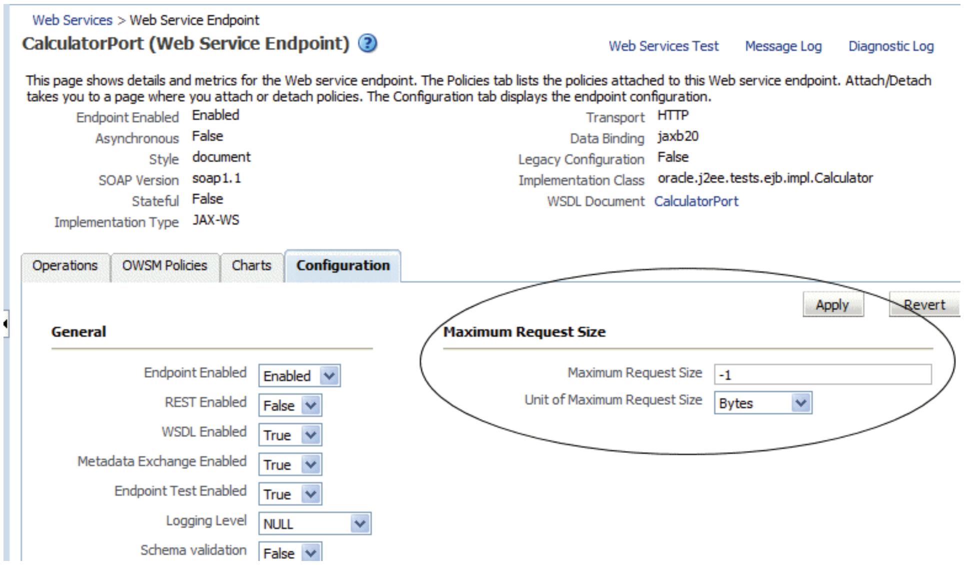
6.17.1 Using Fusion Middleware Control

To set the size of the request message:

1. Navigate to the Web Service Endpoint page, or the Service Home page (for SOA composites), as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click the **Configuration** tab. For SOA composites, click the **Properties** tab.
3. Set the Maximum Request Size and the Unit of Maximum Request Size and click **Apply**.

Note: If you set the Maximum Request Size to -1, indicating that there is no maximum request size, then the Unit of Maximum Request Size setting is irrelevant and defaults to bytes.

Figure 6–8 Setting Size of Request Message



Web Services > Web Service Endpoint

CalculatorPort (Web Service Endpoint)

This page shows details and metrics for the Web service endpoint. The Policies tab lists the policies attached to this Web service endpoint. Attach/Detach takes you to a page where you attach or detach policies. The Configuration tab displays the endpoint configuration.

Endpoint Enabled	Enabled	Transport	HTTP
Asynchronous	False	Data Binding	jaxb20
Style	document	Legacy Configuration	False
SOAP Version	soap1.1	Implementation Class	oracle.j2ee.tests.ejb.impl.Calculator
Stateful	False	WSDL Document	CalculatorPort
Implementation Type	JAX-WS		

General

Endpoint Enabled	Enabled
REST Enabled	False
WSDL Enabled	True
Metadata Exchange Enabled	True
Endpoint Test Enabled	True
Logging Level	NULL
Schema validation	False

Maximum Request Size

Maximum Request Size	-1
Unit of Maximum Request Size	Bytes

-1 sets no limit to the size of the message. Or, you can set a maximum limit to the message by entering a number in the text box and selecting the unit of measurement.

4. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

6.17.2 Using WLST

To set the size of a request message for a Web service endpoint (port), use the `setWebServiceConfiguration` command. Set the `maxRequestSize` property of the `itemProperties` argument to the desired value. Enter a long integer to set the maximum value, or `-1` to set no limit to the size of the message. The default is `-1`.

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Endpoint" on page 6-13.

For example, to specify that there is no message limit size for the `WsdlConcretePort`, use the following command:

```
wls:/wls-domain/serverConfig> setWebServiceConfiguration
("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb","web","{http://namespace/}WsdlC
oncreteService",
"WsdlConcretePort",[("maxRequestSize","-1")])
```

For more information about this WLST command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

6.18 Configuring Asynchronous Web Services

When you invoke a Web service synchronously, the invoking client application waits for the response to return before it can continue with its work. In cases where the response returns immediately, this method of invoking the Web service might be adequate. However, because request processing can be delayed, it is often useful for the client application to continue its work and handle the response later on. By calling a Web service asynchronously, the client can continue its processing, without interrupt, and will be notified when the asynchronous response is returned.

For information about developing asynchronous Web services, see "Developing Asynchronous Web Services" in *Developer's Guide for Oracle Infrastructure Web Services*.

The following procedure describes how to configure your deployed asynchronous Web services. You can also configure asynchronous *Callback client*, as described in "Configuring Asynchronous Web Service Callback Clients" on page 6-30.

To configure asynchronous Web services:

1. Navigate to the Web Services Summary page.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service endpoints if they are not already displayed.
3. Click the name of the endpoint of the asynchronous Web service to navigate to the Web Service Endpoint page.

   For an asynchronous Web service, the Asynchronous flag at the top of the page is set to True. Review the following flags, which provide more information about the asynchronous Web service:

   - Transaction Enabled for Request Queue—Flag that specifies whether transactions are enabled on the request queue.
   - Using Response Queue—Flag that specifies whether a response queue is being used. If set to false, then the response is sent directly to the Web service client, without being stored.
   - Transaction Enabled for Response Queue—Flag that specifies whether transactions are enabled on the response queue.

   These flags are configured at design time. For more information, see "Developing Asynchronous Web Services" in *Developer's Guide for Oracle Infrastructure Web Services*.
4. From the Web Service Endpoint page, click the **Configuration** tab.
5. Under the Asynchronous Web Service section of the page, you can set the configuration properties defined in Table 6–2.

   > **Note:** The configuration properties defined in Table 6–2 appear and are valid only for asynchronous Web services.

Table 6–2 Configuration Properties for Asynchronous Web Services

Configuration Property	Description
JMS Request Queue Connection Factory Name	Name of the connection factory for the JMS request queue. The default JMS connection factory, weblogic.jms.XAConnectionFactory, provided with the base domain is used by default.
JMS Request Queue Name	Name of the request queue. The following queue is used by default: oracle.j2ee.ws.server.async.DefaultRequestQueue.
JMS Response Queue Connection Factory Name	Name of the connection factory for the JMS response queue. The default JMS connection factory, weblogic.jms.XAConnectionFactory, provided with the base domain is used by default.
JMS Response Queue Name	Name of the request queue. The following queue is used by default: oracle.j2ee.ws.server.async.DefaultResponseQueue.
JMS System User	The user that is authorized to use the JMS queues. By default, this property is set to OracleSystemUser. **Note:** For most users, the OracleSystemUser is sufficient. However, if you need to change this user to another user in your security realm, you can do so using the instructions provided in "Changing the JMS System User for Asynchronous Web Services" on page 14-39.

6. Click **Apply**.
7. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

> **Note:** You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

6.19 Enabling and Disabling MTOM

Support for MTOM is provided by attaching the oracle/wsmtom_policy policy to a Web service. You can enable or disable MTOM for a Web service by enabling or disabling this policy. See "Enabling or Disabling a Policy for a Single Policy Subject" on page 7-23 for more information.

You must restart the application after enabling or disabling MTOM.

6.20 Configuring the Web Service Client

> **Note:** The procedures described in this section apply to Oracle Infrastructure Web services only.

For the Web service clients in your application, including SOA references, ADF data control, and asynchronous Web service Callback clients, you can set the configuration properties defined in Table 6–3.

Table 6–3 Configuration Properties for Web Service Clients

Configuration Property	Property Name	Description
General		
UDDI ServiceKey (SOA reference clients only)	oracle.soa.uddi.serviceKey	Specifies the service key of the Oracle Service Registry (OSR) if UDDI is used for run-time resolution of the endpoint. For more information, see "Changing the Endpoint Reference and Service Key for Oracle Service Registry Integration" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.
Endpoint Address	javax.xml.ws.service.endpoint.address	Endpoint URL to which the client will send the request. **Note**: This property is not available for asynchronous Web service Callback clients.
Maintain Session	javax.xml.ws.session.maintain	Flag that specifies whether the session should be maintained. **Note**: This property is not available for asynchronous Web service Callback clients.
Atomic Transaction Version (SOA reference clients only)	wsat.Version	Specifies the version of the SOA Web service atomic transaction coordination context used for outbound messages only. The value specified must be consistent across the entire transaction. Valid values are **WSAT10**, **WSAT11**, **WSAT12**, and **Default**. Note that if the flow option is set to **WSDL Driven**, you cannot specify a version. The version advertised in the WSDL is used. If the flow option is set to **Supports** or **Mandatory** and you specify the **Default** option, then WSAT10 is used. **Note:** In WLST, the valid values must be specified as "WSAT10", "WSAT11", "WSAT12", and "DEFAULT". Use of an invalid value results in an error message.

Table 6–3 (Cont.) Configuration Properties for Web Service Clients

Configuration Property	Property Name	Description
Atomic Transaction Flow Option (SOA reference clients only)	wsat.flowOption	Specifies whether the transaction coordination context is passed with the transaction flow. Valid values on the SOA reference client are: ■ **Never** (default) – Do not export transaction coordination context. ■ **Supports** – Export transaction coordination context if transaction is available. ■ **Mandatory** – Export transaction coordination context. An exception is thrown if there is no active transaction. ■ **WSDL Driven** – Use the value set in the WSDL. **Note:** In WLST, the valid values must be specified as "NEVER", "SUPPORTS", "MANDATORY", and "WSDLDriven". Use of an invalid value results in an error message.
HTTP Chunking		
Stop Chunking	oracle.webservices.donotChunk	Flag that specifies whether chunking is enabled for client requests.
Chunking Size (bytes)	oracle.webservices.chunkSize	Size of the request chunk in bytes.
HTTP Timeout		
HTTP Read Timeout (ms)	oracle.webservices.httpReadTimeout	Length of the request read timeout in milliseconds.
HTTP Connection Timeout (ms)	oracle.webservices.httpConnTimeout	Length of the request connection timeout in milliseconds.
HTTP Basic Authentication		
HTTP User Name	(javax.xml.ws.security.auth.username) oracle.webservices.auth.username	Authenticated HTTP user name.
HTTP User Password	(javax.xml.ws.security.auth.password) oracle.webservices.auth.password	Authenticated HTTP user password.
Preemptive	oracle.webservices.preemptiveBasicAuth	Flag that specifies whether security will be sent with the request without being challenged.
HTTP Proxy		
Proxy Host	oracle.webservices.proxyHost	URL of proxy to which client will send the request.
Proxy Port	oracle.webservices.proxyPort	Port number of the proxy.
Proxy User Name	oracle.webservices.proxyUsername	Valid user name to access the proxy.
Proxy User Password	oracle.webservices.proxyPassword	Valid password to access the proxy.
Proxy Realm	oracle.webservices.proxyAuthRealm	Realm used by the proxy.
Proxy Authentication Type	oracle.webservices.proxyAuthType	Authentication type used by the proxy.

The following sections describe how to configure Web service clients using Fusion Middleware Control and WLST.

6.20.1 Using Fusion Middleware Control

The following procedures describe how to configure SOA reference, ADF DC, WebCenter, and asynchronous Web service Callback clients.

6.20.1.1 Configuring SOA References

The following procedure describes how to configure a SOA reference.

1. View the SOA reference, as described in "Viewing SOA References" on page 6-10.
2. Click the **Properties** tab.
3. Set the property values as required. Refer to Table 6–3.
4. Click **Apply**.

6.20.1.2 Configuring ADF DC Web Service Clients

The following procedure describes how to configure an ADF DC Web service client.

1. View the ADF DC Web service client, as described in "Viewing Connection-Based Web Service Clients" on page 6-10.
2. Click the **Configuration** tab.
3. Set the configuration values as required. Refer to Table 6–3.
4. Click **Apply**.

6.20.1.3 Configuring Asynchronous Web Service Callback Clients

The following procedure describes how to configure an asynchronous Web service Callback client. Callback clients are used only by asynchronous Web services to return the response to the caller. For more information, see "Developing Asynchronous Web Services" in *Developer's Guide for Oracle Infrastructure Web Services*.

To configure an asynchronous Web service Callback client:

1. Navigate to the endpoint for the asynchronous Web service, as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click **Callback Client** in the upper right portion of the endpoint page.
3. Click the **Configuration** tab.
4. Set the configuration values as required. Refer to Table 6–3.
5. Click **Apply**.

6.20.2 Using WLST

Use the following procedure to configure the Web service client endpoint (port) using WLST:

1. Connect to the running instance of WebLogic Server to which the application is deployed as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Use the `listWebServiceClients` WLST command to display a list of the Web service clients in your application as described in "Viewing Web Service Clients" on page 6-9.

3. Use the `listWebServiceClientPorts` command to display the endpoint name and endpoint URL for a Web service client.

   ```
   listWebServiceClientPorts(application,moduleOrCompName,moduleType,serviceRefNam
   e)
   ```

   For example, to display the endpoint for the service reference `client`:

   ```
   wls:/wls-domain/serverConfig> listWebServiceClientPorts('/base_
   domain/AdminServer/application1#V2.0',
   'test1','wsconn','client')

   HelloWorld_pt
   ```

4. Use the `listWebServiceClientStubProperties` command to view the configuration details for a Web service client endpoint.

   ```
   listWebServiceClientStubProperties(application, moduleOrCompName, moduleType,
   serviceRefName,portInfoName)
   ```

   For example, to view the configuration details for the `HelloWorld_pt`:

   ```
   wls:/wls-domain/serverConfig> listWebServiceClientStubProperties('/base_
   domain/AdminServer/application1#V2.0',
   'test1','wsconn','client','HelloWorld_pt')

   keystore.recipient.alias=A1
   saml.issuer.name=B1
   user.roles.include=C1
   ```

   Alternatively, you can set the `detail` argument to `true` in the `listWebServiceClients` command to view the configuration details for the endpoint as shown in "Using WLST" in "Viewing Web Service Clients" on page 6-9.

5. Do one of the following:

   - Use the `setWebServiceClientStubProperty` command to set or change a single stub property of a Web service client endpoint. Specify the property to be set or changed using the `propName` and `propValue` arguments. To remove a property, specify a blank value for the `propValue` argument.

     ```
     setWebServiceClientStubProperty(application,moduleOrCompName,moduleType,
      serviceRefName,portInfoName,propName,[propValue])
     ```

     For example, to change the `keystore.recipient.alias` to `oracle` for the `HelloWorld_pt`, use the following command:

     ```
     wls:/wls-domain/serverConfig> setWebServiceClientStubProperty('/base_
     domain/AdminServer/application1#V2.0',
     'test1','wsconn','client','HelloWorld_
     pt','keystore.recipient.alias','oracle')
     ```

   - Use the `setWebServiceClientStubProperties` command to configure the set of properties of a Web service client endpoint. Specify the properties to be set or changed using the `properties` argument.

     ```
     setWebServiceClientStubProperties(application, moduleOrCompName,
      moduleType, serviceRefName, portInfoName, properties)
     ```

     This command configures or resets all of the stub properties for the Oracle WSM client security policy attached to the client. Each property that you list in the command is set to the value you specify. If a property that was previously set is not explicitly specified in this command, it is reset to the default for the property. If no default exists, the property is removed.

For example, to configure atomic transactions for the `TaskReference_pt` SOA reference endpoint of the `default/SimpleRef[1.0]` SOA composite application, use the following command:

```
wls:soainfra/serverConfig>
 setWebServiceClientStubProperties('soa-infra','default/SimpleRef[1.0]',
'soa','client', 'TaskReference_pt',
[("wsat.flowOption","SUPPORTS"),("wsat.Version","DEFAULT")])
```

To verify that the reference is properly configured, enter the following command:

```
wls:soainfra/serverConfig>listWebServiceClients(None, None, true)

    /soainfra/soa_server1/soa-infra:
        compositeName=default/SimpleRef[1.0], moduleType=soa,
serviceRefName=client
                TaskReference_pt
                wsat.version=DEFAULT
                wsat.flowOption=SUPPORTS
```

For more information about the client properties that you can set, see Table 6–3, " Configuration Properties for Web Service Clients". When specifying these properties, use the format shown in the Property Name column.

You can also set the properties described in "Attaching Client Policies Permitting Overrides" on page 8-31.

6. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

For more information about these WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

7 Managing Web Service Policies

This chapter includes the following sections:

7.1 Overview of Web Services Policy Management

For information about Web services policies and how Oracle Fusion Middleware uses policies to manage Quality of Service (QoS) for Web services, see "Understanding Oracle WSM Policy Framework" on page 3-1."

7.2 Viewing Available Web Services Policies

You can use both Fusion Middleware Control and the WebLogic Scripting Tool (WLST) to view the Web service polices in your domain. In Fusion Middleware Control, you view the policies using the Web Services Policies page.

Use the procedures in the following sections to view a list of the policies.

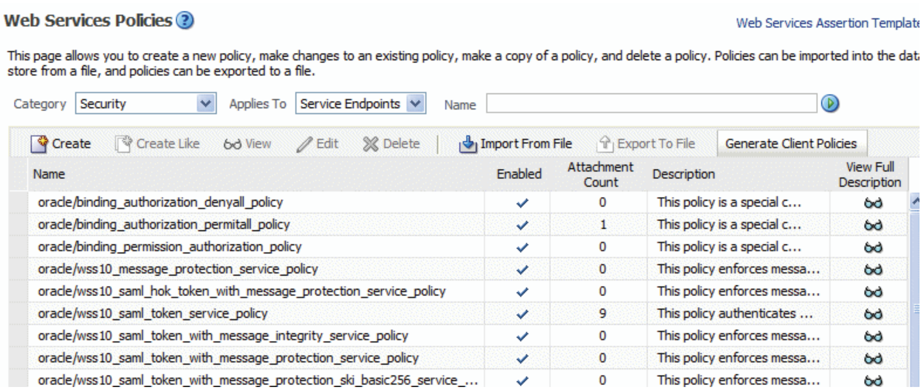
7.2.1 Navigating to the Web Services Policies Page in Fusion Middleware Control

You manage the Web services policies in your farm from the Web Services Policies page. From this page, you can view, create, edit, and delete Web services policies.

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you want to see the policies. Select the domain.
2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Web Services** and then **Policies**.

   The Web Services Policies page is displayed (Figure 7–1).

Figure 7–1 Web Services Policy Page



This page allows you to create a new policy, make changes to an existing policy, make a copy of a policy, and delete a policy. Policies can be imported into the dat store from a file, and policies can be exported to a file.

Category: Security
Applies To: Service Endpoints
Name:

Name	Enabled	Attachment Count	Description	View Full Description
oracle/binding_authorization_denyall_policy	✓	0	This policy is a special c...	👓
oracle/binding_authorization_permitall_policy	✓	1	This policy is a special c...	👓
oracle/binding_permission_authorization_policy	✓	0	This policy is a special c...	👓
oracle/wss10_message_protection_service_policy	✓	0	This policy enforces messa...	👓
oracle/wss10_saml_hok_token_with_message_protection_service_policy	✓	0	This policy enforces messa...	👓
oracle/wss10_saml_token_service_policy	✓	9	This policy authenticates ...	👓
oracle/wss10_saml_token_with_message_integrity_service_policy	✓	0	This policy enforces messa...	👓
oracle/wss10_saml_token_with_message_protection_service_policy	✓	0	This policy enforces messa...	👓
oracle/wss10_saml_token_with_message_protection_ski_basic256_service_...	✓	0	This policy enforces messa...	👓

7.2.2 Displaying a List of the Available Policies Using WLST

To display a list of the available policies using WLST:

1. Connect to the running instance of WebLogic Server for which you want to view the Web services as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Use the `listAvailableWebServicePolicies()` WLST command to display a list of the Web services.

   ```
   listAvailableWebServicePolicies([category],[subject])
   ```

   For example:

   ```
   wls:/base_domain/domainRuntime> listAvailableWebServicePolicies()

   List of available OWSM policy - total : 58
   security : oracle/binding_authorization_denyall_policy
   security : oracle/binding_authorization_permitall_policy
   security : oracle/binding_permission_authorization_policy
   security : oracle/component_authorization_denyall_policy
   security : oracle/component_authorization_permitall_policy
   security : oracle/component_permission_authorization_policy
   management : oracle/log_policy
   addressing : oracle/wsaddr_policy
   mtom : oracle/wsmtom_policy
   wsrm : oracle/wsrm10_policy
   wsrm : oracle/wsrm11_policy
   ```

3. Use the optional `category` and `subject` arguments to specify the policy category, such as security or management, and the policy subject type, such as server or client.

   For example:

```
wls:/base_domain/domainRuntime>
listAvailableWebServicePolicies("security","server")
List of available OWSM policy - total : 39
security : oracle/wss_saml_or_username_token_service_policy
security : oracle/wss10_username_token_with_message_protection_service_policy
security : oracle/wss10_x509_token_with_message_protection_service_policy
security : oracle/no_messageprotection_service_policy
security : oracle/wss_saml_or_username_token_over_ssl_service_policy
security : oracle/wss10_username_token_with_message_protection_ski_basic256_
service_policy
security : oracle/wss11_saml20_token_with_message_protection_service_policy
security : oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy
security : oracle/wss11_sts_issued_saml_hok_with_message_protection_service_
policy
security : oracle/wss11_kerberos_token_service_policy
```

7.3 Viewing a Web Service Policy

Follow the procedure below to view the policy details in read-only mode.

To view a Web service policy

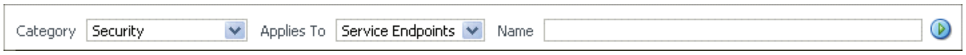
1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select a policy from the Policies table and click **View.**
3. When you are done viewing the policy, click **Return to Web Services Policies**.

7.4 Searching for Web Service Policies

In the Web Services Policies page, you can narrow down the number of policies that are returned by specifying criteria in the Search Filter (Figure 7–2).

The wildcard character asterisk (*) in the Name field matches any characters.

Figure 7–2 Search Filter Criteria



Category: Security
Applies To: Service Endpoints
Name:

The policies that are returned are those that match the criteria specified in the Category, Applies To, and Name fields (Table 7–1).

Table 7–1 Search Filter Criteria

Field	Description
Category	Category to which the Web service policy belongs. The options are: - All - Security - MTOM Attachments - Reliable Messaging - WS-Addressing - Management
Applies To	Policy subject to which the policy can be attached. The options are: - **All** – All means that the policy is targeted for any type of endpoint. All refers to the policies that can be applied to Service Endpoints, or Service Clients, or SOA Components. - **Service Endpoints** – Policies that can be attached to Web services. See "Types of Web Services and Clients" in *Oracle Fusion Middleware Introducing Web Services*. - **Service Clients** – Policies that can be attached to Web service clients. See "Types of Web Services and Clients" in *Oracle Fusion Middleware Introducing Web Services*. - **SOA Components** – Policies that can be attached to SOA components SOA Web services are categorized as Service Endpoints, and SOA references are categorized as Service Clients.
Name	Name of the policy. You can enter the complete name or part of policy name. For example, if you enter *http*, any policy with *http* in any part of its name is returned.

For example, if *Security* is selected in the **Category** field, and *Service Endpoints* is selected in the **Applies To** field, and the **Name** field is left blank, then the policies returned are those security policies that can be attached to Web service endpoints.

7.5 Creating Web Service Policies

You can create a Web service policy in one of the following ways:

- Creating a new policy using assertion templates
- Creating a policy from an existing policy
- Importing a policy from a file
- Creating custom policies

The sections that follow describe how to create policies using each of these methods.

7.5.1 Creating a New Web Service Policy

Follow the procedure below to create a new policy using one or more assertion templates.

To create a new Web service policy

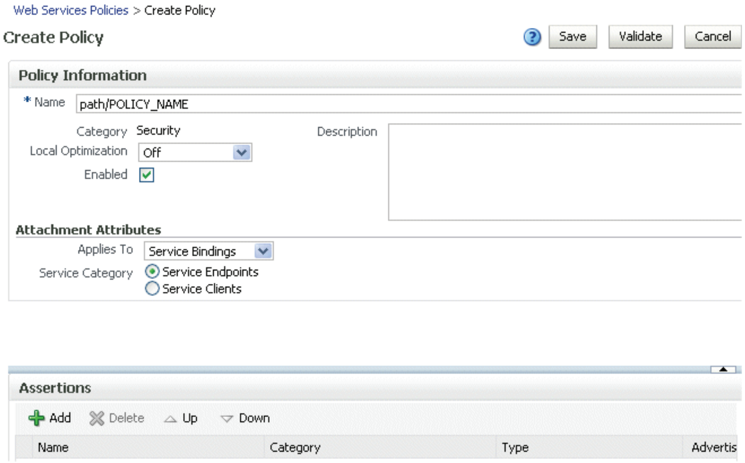
1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the **Category** menu, select the category to which this policy will belong and click **Create**.

> **Note:** The **Create** button is available only for the Security and Management categories.

3. In the Create Policy page (Figure 7–3), enter the path, name, and brief description for your policy. All policies are identified by the directory in which the policy is located.

   Oracle recommends that you follow the policy naming conventions described in "Recommended Naming Conventions for Policies" on page 3-10.

Figure 7–3 Create Policy Page



Policy Information

* Name: path/POLICY_NAME

Category: Security

Local Optimization: Off

Enabled: ☑

Description:

Attachment Attributes

Applies To: Service Bindings

Service Category: ● Service Endpoints ○ Service Clients

Assertions

Name	Category	Type	Advertis

> **Note:** You cannot edit the name of a policy once the policy is created. To change the policy name, you will need to *copy* the policy and assign it a different name.

4. Set the **Local Optimization** control. See "Configuring Local Optimization for a Policy" on page 11-142 for a description of the Local Optimization control.
5. By default, the policy is enabled. If you want to disable the policy, clear the **Enabled** box. A policy that is not enabled is not enforced at run time.
6. Specify the type of policy subjects the policy can be attached to by selecting from the **Applies To** menu. If you select Service Bindings, then specify whether the policy can be attached to Web service endpoints, Web service clients, or to both.

   Of the predefined assertions, only assertions (which you add next) of type security/logging can be added under Service Category *Both*. If you plan to add other types of assertions, choose *Service Endpoints* or *Service Clients*.

7. To add a single assertion:

   a. In the Assertions section, click **Add**.

- b. In the Add Assertion box, enter a meaningful name for your assertion, and select an assertion template from the **Assertion Template** list.

  See Appendix C, "Predefined Assertion Templates" for information on the Oracle Fusion Middleware Web Services policy assertion templates.

- c. Click **OK**.

8. To add an OR group, click **Add OR Group**. For more details, see "Adding an OR Group to a Policy" on page 7-13.
9. In the Assertions section, select the assertion you just added.
10. In the Assertion Details section, enter a description for the assertion.
11. If active for the assertion category, on the **Settings** tab specify the properties for the assertion. Click the **Help** icon for information on setting the properties.
12. If active for the assertion category, click the **Configurations** tab to set the configuration options. Click the **Help** icon for information on setting the properties.
13. Add additional assertions as needed.
14. When you have finished adding assertions, select the assertions and use the **Up** and **Down** controls to order them as needed. Assertions are invoked in the order in which they appear in the list.
15. Click **Validate** to verify that the policy does not contain errors. For more information on policy validation, see "Validating Web Services Policies" on page 7-15.

    If the policy is invalid, it is disabled as a precaution. After you correct the validation issues, you will have to enable the policy.

16. Click **Save.**

7.5.2 Creating a Web Service Policy from an Existing Policy

You can take a Web service policy and use it as a base for creating another policy. By default, Oracle WSM delivers a set of predefined policies. You can create a copy of one of the predefined policies or you can create a copy of a policy that you have created. Once the policy is created, you can treat it like any other policy, adding or deleting assertions, and modifying existing assertions.

To make a copy of a Web service policy

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select a policy from the Policies list and click **Create Like.**
3. In the Create Policy page, enter a name for the policy.

   The word *Copy* is appended to the name of the copied policy and, by default, this is the name assigned to the new policy. For example, if the policy being copied is named *oracle/wss10_username_token_service*, then the default name of the copy is *oracle/wss10_username_token_service_Copy.*

   It is recommended that you change the name of this new policy to be more meaningful in your environment.

4. Modify the policy as required, including the assertions.

5. Click **Validate** to verify that the policy does not contain errors. For more information on policy validation, see "Validating Web Services Policies" on page 7-15.
6. Click **Save.**

7.5.3 Importing Web Service Policies

Follow the procedure in this section to import a policy into the Oracle WSM Repository. Once the policy is imported, you can attach it to Web services and make changes to it.

> **Note:** The policy name you import must not already exist in the repository.
>
> Be aware that "policy name" and "file name" are different. The policy name is specified by the name attribute of the policy content; the file name is the name of the policy file. You might find it convenient for the two names to match, but it is not required.
>
> You cannot prefix the name of a policy with oracle_. Otherwise, you will receive exceptions when you try to use the policy.

To import a Web service policy

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, click **Import From File.**
3. In the Create Policy From File box, enter the file path of the file in the Select Policy File Box. Or, you can click on the Browse button and select the policy file.
4. Click **OK**.

7.5.4 Creating Custom Policies

For information about creating custom Web service policies using custom assertions, see "Creating Custom Assertions" in *Extensibility Guide for Oracle Web Services Manager*.

7.6 Managing Policy Assertion Templates

Your Fusion Middleware installation includes predefined assertion templates that you can use to construct your policies or copy to create new policies. For additional information, see "Building Policies Using Policy Assertions" on page 3-5.

You can add one or more assertions to a policy. The predefined assertions are described in Appendix C, "Predefined Assertion Templates". Assertions are executed in the order in which they appear in the list. You can change the order of the assertions in the list by selecting the assertion and clicking the **Up** or **Down** arrow.

The following sections provide more information about working with assertions:

- "Navigating to the Web Services Assertion Templates Page" on page 7-8
- "Naming Conventions for Assertion Templates" on page 7-8
- "Viewing an Assertion Template" on page 7-9
- "Searching for an Assertion Template" on page 7-9

- "Creating an Assertion Template" on page 7-9
- "Editing an Assertion Template" on page 7-10
- "Editing the Configuration Properties" on page 7-11
- "Adding Assertions to a Policy" on page 7-12
- "Adding an OR Group to a Policy" on page 7-13
- "Configuring Assertions" on page 7-14
- "Exporting an Assertion Template" on page 7-14
- "Importing an Assertion Template" on page 7-15
- "Deleting an Assertion Template" on page 7-15

7.6.1 Navigating to the Web Services Assertion Templates Page

You can manage your assertion templates at the domain level from the Web Services Assertion Template page. From this page, you can copy, edit, and delete assertion templates.

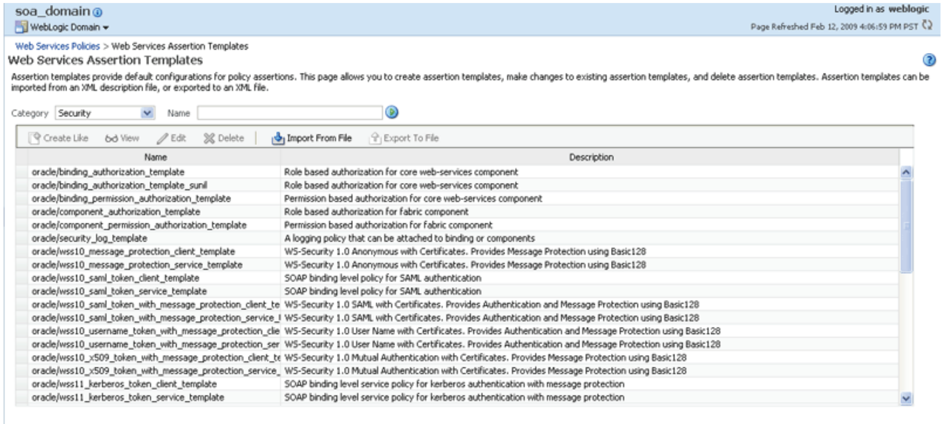
To navigate to the Web Services Assertion Templates page:

1. In the Navigator pane, expand **WebLogic Domain**.
2. Click the domain for which you want to manage assertion templates.
3. From the WebLogic Domain menu select **Web Services > Policies**.

   The Web Services Policies page is displayed.
4. Click **Web Services Assertion Templates** in the upper right corner of the page.

   The Web Services Assertion Templates page is displayed, as shown in the following figure.

Figure 7–4 Web Services Assertion Templates Page



soa_domain
WebLogic Domain
Logged in as weblogic
Page Refreshed Feb 12, 2009 4:06:59 PM PST
Web Services Policies > Web Services Assertion Templates

Web Services Assertion Templates

Assertion templates provide default configurations for policy assertions. This page allows you to create assertion templates, make changes to existing assertion templates, and delete assertion templates. Assertion templates can be imported from an XML description file, or exported to an XML file.

Category: Security

Name:

Name	Description
oracle/binding_authorization_template	Role based authorization for core web-services component
oracle/binding_authorization_template_sunil	Role based authorization for core web-services component
oracle/binding_permission_authorization_template	Permission based authorization for core web-services component
oracle/component_authorization_template	Role based authorization for fabric component
oracle/component_permission_authorization_template	Permission based authorization for fabric component
oracle/security_log_template	A logging policy that can be attached to binding or components
oracle/wss10_message_protection_client_template	WS-Security 1.0 Anonymous with Certificates. Provides Message Protection using Basic128
oracle/wss10_message_protection_service_template	WS-Security 1.0 Anonymous with Certificates. Provides Message Protection using Basic128
oracle/wss10_saml_token_client_template	SOAP binding level policy for SAML authentication
oracle/wss10_saml_token_service_template	SOAP binding level policy for SAML authentication
oracle/wss10_saml_token_with_message_protection_client_te	WS-Security 1.0 SAML with Certificates. Provides Authentication and Message Protection using Basic128
oracle/wss10_saml_token_with_message_protection_service_I	WS-Security 1.0 SAML with Certificates. Provides Authentication and Message Protection using Basic128
oracle/wss10_username_token_with_message_protection_clie	WS-Security 1.0 User Name with Certificates. Provides Authentication and Message Protection using Basic128
oracle/wss10_username_token_with_message_protection_ser	WS-Security 1.0 User Name with Certificates. Provides Authentication and Message Protection using Basic128
oracle/wss10_x509_token_with_message_protection_client_te	WS-Security 1.0 Mutual Authentication with Certificates. Provides Message Protection using Basic128
oracle/wss10_x509_token_with_message_protection_service_	WS-Security 1.0 Mutual Authentication with Certificates. Provides Message Protection using Basic128
oracle/wss11_kerberos_token_client_template	SOAP binding level service policy for kerberos authentication with message protection
oracle/wss11_kerberos_token_service_template	SOAP binding level service policy for kerberos authentication with message protection

7.6.2 Naming Conventions for Assertion Templates

The same naming conventions used to name predefined policies are used to name the assertion templates. Assertion templates begin with the directory name *oracle/* and are identified with the suffix *_template* at the end; for example, *oracle/wss10_message_protection_service_template*. For more information on naming conventions for predefined policies, see "Recommended Naming Conventions for Policies" on page 3-10.

7.6.3 Viewing an Assertion Template

Follow the steps in this section to view an assertion template.

1. Navigate to the Web Services Assertion Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 7-8.
2. From the table, select the assertion template that you want to view.
3. Click **View**.
4. In the View Template page, review the assertion.
5. When you are done, click **Return to Web Services Assertion Templates.**

7.6.4 Searching for an Assertion Template

You can search for a Web service assertion template by category, name, or both. To do so:

1. Navigate to the Web Services Assertion Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 7-8.
2. Perform one or more of the following steps:
   - To search for assertion templates in a specific category (or all categories), select a category from the Category list.

     Valid categories include: All, Security, MTOM Attachments, Reliable Messaging, WS-Addressing, and Management.
   - To search for an assertion template that contains a specific string, enter a string in the **Name** field.

     Specify any portion of the name of an assertion template to display all assertion templates that contain the string for the specified category.
3. Click the Search Assertion Templates icon next to the **Name** field.

   The assertion templates list is refreshed to include only those assertion templates that match the specified search criteria.

7.6.5 Creating an Assertion Template

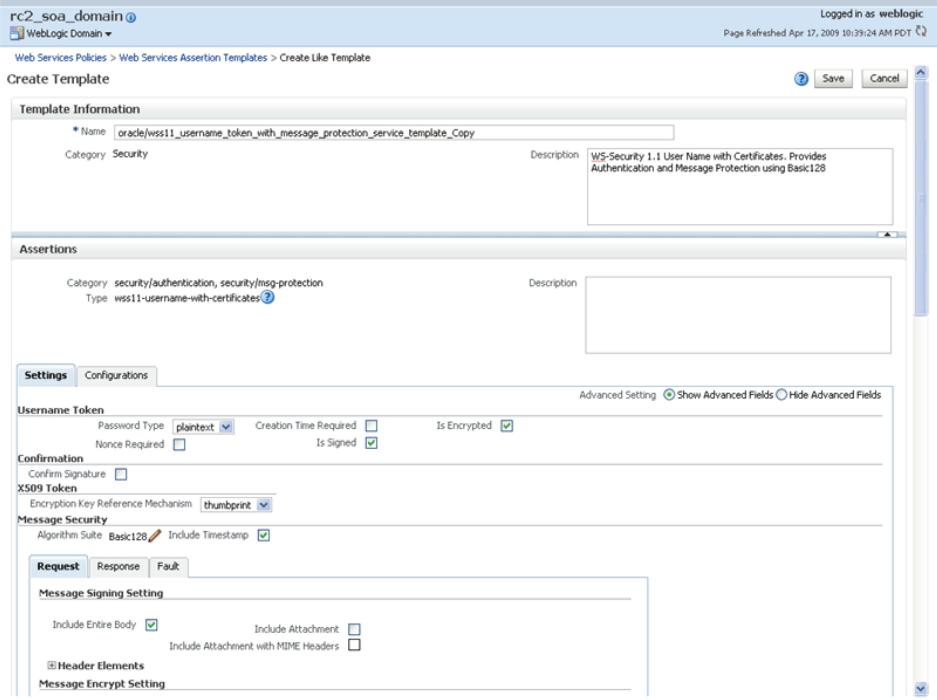
A new assertion template is created based on an existing assertion. Pick the assertion template that most closely matches the desired behavior, then make any changes required to get the new behavior.

To create an assertion template:

1. Navigate to the Web Services Assertion Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 7-8.
2. Select the assertion template from the Assertion Templates table that you want to copy.
3. Click **Create Like**.

   The following graphic shows the Create Template page.

Figure 7–5 Create Template Page



rc2_soa_domain
WebLogic Domain
Logged in as weblogic
Page Refreshed Apr 17, 2009 10:39:24 AM PDT

Web Services Policies > Web Services Assertion Templates > Create Like Template

Create Template

Template Information

* Name: oracle/wss11_username_token_with_message_protection_service_template_Copy
Category: Security
Description: WS-Security 1.1 User Name with Certificates. Provides Authentication and Message Protection using Basic128

Assertions

Category: security/authentication, security/msg-protection
Type: wss11-username-with-certificates
Description:

Advanced Setting ● Show Advanced Fields ○ Hide Advanced Fields

Username Token
Password Type: plaintext
Creation Time Required □
Is Encrypted ☑
Nonce Required □
Is Signed ☑

Confirmation
Confirm Signature □

X509 Token
Encryption Key Reference Mechanism: thumbprint

Message Security
Algorithm Suite: Basic128
Include Timestamp ☑

Message Signing Setting
Include Entire Body ☑
Include Attachment □
Include Attachment with MIME Headers □
⊞ Header Elements

Message Encrypt Setting

4. In the Template Information section, edit the name of the assertion and, optionally, enter a brief description.

   The word *Copy* is appended to the name of the copied assertion template and, by default, this is the name assigned to the new assertion template. For example, if the assertion template being copied is named *oracle/wss10_username_token_service_template,* then the default name of the copy is *oracle/wss10_username_token_service_template_Copy.*

   It is recommended that you change the name of this new assertion template to be more meaningful in your environment.

5. Click **Save.**

   The assertion is added to the Assertion Templates table. You can now select the new assertion and click **Edit** to configure the assertion.

7.6.6 Editing an Assertion Template

Note: Oracle recommends that you do not edit the predefined assertion templates so that you will always have a known set of valid templates.

Follow the steps in this section to edit an assertion template.

1. Navigate to the Web Services Assertions Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 7-8.
2. From the table, select the assertion template that you want to edit.
3. Click **Edit**.

4. Click the **Settings** or **Configuration** tabs and edit the assertion template as required.

   The settings that can be edited for each template are described in Appendix C, "Predefined Assertion Templates.". For information about the properties that you can edit from the **Configuration** tab, see "Editing the Configuration Properties" on page 7-11.

5. When you are finished editing the template, click **Save**.

7.6.7 Editing the Configuration Properties

Predefined security assertion templates include configuration properties that you can configure to match your environment. For example, properties that are configurable in assertion templates include `csf-key`, `saml.issuer.name`, `keystore.recipient.alias`, and `role`, among others. When you edit an existing predefined assertion template or create an assertion template using the **Create Like** option in Fusion Middleware Control, you can configure the following settings for each property:

> **Note:** Oracle recommends that you do not edit the predefined assertion templates so that you will always have a known set of valid templates.

- **Description**—Description of the property.
- **Value**—Current value.
- **Default**—Default value. This value is used if the **Value** field is not set.
- **Content Type**—Can be one of the following:
  - **Constant**—Property cannot be overridden.
  - **Required**—Property is required and can be overridden.
  - **Optional**—Property is optional and can be overridden.

To configure the properties:

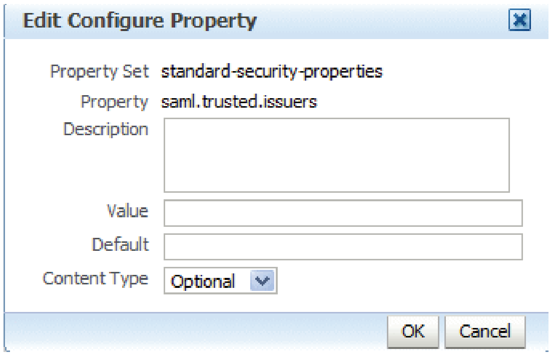
1. Select the assertion template to be edited as described in "Editing an Assertion Template" on page 7-10.
2. Click the **Configurations** tab.

   The list of properties for the template are displayed.

3. Select the property from the list and click **Edit**.

   The Edit Configure Property box is displayed, as shown in Figure 7–6.

Figure 7–6 Edit Configure Property Window Displayed When Creating an Assertion



Edit Configure Property

Property Set standard-security-properties

Property saml.trusted.issuers

Description

Value

Default

Content Type Optional

4. Enter the values for your configuration and click **OK**.

Note: When you add an assertion to a policy, as described in "Adding Assertions to a Policy" on page 7-12, you can set the assertion configuration properties, specifically the **Value**, **Default**, and **Description** properties, to match your environment. The **Content Type** property setting defined in the assertion template cannot be changed, and is not displayed in the Edit Configure Property window.

7.6.8 Adding Assertions to a Policy

You can add assertions from the Create Policy page, the Copy Policy page, or the Edit Policy Detail page.

Each policy can contain only one assertion for each of the following categories: MTOM Attachments and Reliable Messaging. The policy can contain any number of assertions belonging to the Security category; however, the combination of assertions must be valid. For more information on valid assertions, see "Validating Web Services Policies" on page 7-15.

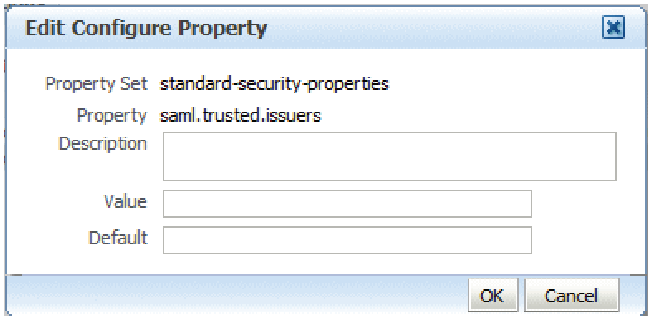
To add an assertion to a policy:

1. Navigate to the Create Policy page, the Create Like page, or the Edit Policy Detail page.
2. In the Assertions section, click **Add**.
3. In the Add Assertion box, enter the name for your assertion, and select an assertion from the Assertion Template list.
4. Click **OK**.
5. To configure the assertion, click the **Settings** tab and edit the settings as required.
6. To edit the configuration properties, click the **Configurations** tab.

   The list of configuration properties defined for the assertion are displayed.
7. Select the property to be edited and click **Edit**.

   The Edit Configure Property window is displayed as shown in Figure 7–7.

Figure 7–7 Edit Configure Property Window Displayed When Displayed in a Policy



Edit Configure Property

Property Set standard-security-properties

Property saml.trusted.issuers

Description

Value

Default

8. Edit the Configuration properties and click **OK**.

   Note that you can edit only the **Description**, **Value**, and **Default** properties. The **Content Type** property setting defined in the assertion template cannot be changed, and is not displayed. For details about these properties, see "Editing the Configuration Properties" on page 7-11.

9. When you are done, click **Save** to save the policy.

7.6.9 Adding an OR Group to a Policy

You can create an OR group, consisting of one or more assertions, enabling a single policy to accept multiple types of security tokens. A client can enforce *any one* of the policies that are defined in the OR group. For more information, see "Defining Multiple Policy Alternatives (OR Groups)" on page 3-8.

You can add only one OR group to a policy. Once you have generated an OR Group, the Add OR Group button is greyed out.

You can add an OR group from the Create Policy page, the Copy Policy page, or the Edit Policy Detail page.

To add an OR group to a policy:

1. Navigate to the Create Policy page, the Create Like page, or the Edit Policy Detail page.
2. In the Assertion List section, click **Add OR Group**.
3. In the Add OR Group dialog, enter the name of the first assertion in the group, and select an assertion template from the Assertion Template list.
4. Click **OK**.

   The assertion is added under the OR Group.

5. To add additional assertions to the OR group:

   a. Ensure that an assertion within the OR group is currently selected.

   b. Click **Add**.

   c. In the Add Assertion dialog, enter the name of the assertion in the group, and select an assertion template from the Assertion Template list.

   d. Click **OK**.

6. To configure the assertions, see "Configuring Assertions" on page 7-14.

   The policy attribute values for attachTo and category limit the assertions that are valid within the current policy. All assertions within an OR group must be compatible with the attachTo and category attribute values in order to be considered.

7. When you have finished adding assertions to the OR group, select the assertions and use the **Up** and **Down** controls to order them as needed. Assertions are considered for invocation in the order that they appear on the list.

8. To delete an assertion from the OR group, select the assertion and click **Delete**. To delete the entire OR group, select the OR group and click **Delete**.

7.6.10 Configuring Assertions

Once an assertion has been added to a policy, you can configure the assertion attributes. You can configure assertions from the Create Policy page, the Create Like page, or the Edit Policy Detail page.

To configure an assertion:

1. Navigate to the Create Policy page, the Create Like page, or the Edit Policy Detail page.

2. In the Assertions section of the page, select the assertion to be configured in the assertion table.

3. Click the **Settings** or **Configurations** tab.

4. Edit the Settings and Configuration properties, and click **Save.**

See Appendix C, "Predefined Assertion Templates" for more information about the Settings and Configuration assertion properties. For information about the configuration properties displayed on the **Configurations** tab, see "Editing the Configuration Properties" on page 7-11.

7.6.11 Exporting an Assertion Template

You can export individual assertion templates from Oracle Enterprise Manager Fusion Middleware Control. You can then copy the assertion template to a directory or import the assertion template to move it to another repository. Once moved, you can import the assertion template, as described in "Importing an Assertion Template" on page 7-15.

To export an assertion template:

1. Navigate to the Web Services Assertions Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 7-8.

2. Select the assertion template from the Assertion Templates table that you want to export to a file.

3. Click **Export to File**.

   You are prompted to open or save the file.

4. Select **Save File**.

5. Click **Ok**.

6. Navigate to the location on your local directory to which you want to save the file and update the filename as desired.

7. Click **Save**.

7.6.12 Importing an Assertion Template

Follow the steps in this section to view an assertion template.

1. Navigate to the Web Services Assertions Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 7-8.
2. Click **Import From File**.

   You are prompted to provide the assertion template file.
3. Click **Browse** to navigate to the directory where the assertion template file is located and select the assertion template to be imported.
4. Click **OK**.

   The assertion template appears in the Assertion Templates table.

7.6.13 Deleting an Assertion Template

Follow the steps in this section to delete an assertion template.

1. Navigate to the Web Services Assertions Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 7-8.
2. Select the assertion template from the Assertion Templates table that you want to delete.
3. Click **Delete**.

   You are prompted to confirm that you want to delete the assertion template.
4. Click **OK.**

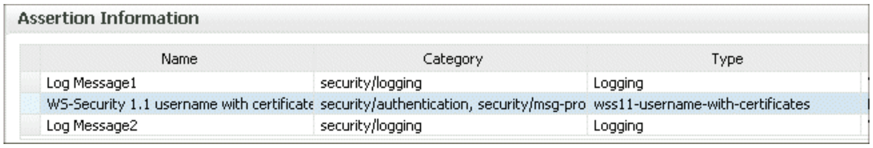
7.7 Validating Web Services Policies

There are restrictions on the type and number of policy assertions that are permitted in a Web service policy. When you validate a policy, Enterprise Manager checks to see if the policy is consistent with these restrictions. A policy can contain only assertions that belong to a single category. Therefore, you cannot combine a Security assertion with an MTOM assertion in the same policy. The policy type is determined by the category of the assertion. Therefore, a policy containing a security assertion is a security policy, a policy containing a management assertion is a management policy, and so on. Security assertions are further categorized into subcategories: authentication, logging, message protection (msg-protection), and authorization.

There are restrictions on the number and type of assertions you can have in a policy. The restrictions are as follows:

- MTOM and Reliable Messaging policies can contain only one assertion.
- A security policy can contain multiple security assertions; however, there can be only one assertion from the following subcategories in a policy: encryption, signing, and authentication.
- Some assertions contain both authentication and message protection. For example, if you view the *oracle/wss11_username_token_with_message_protection_service_policy*, you will see that the second assertion falls into two categories: security/authentication and security/msg-protection.   See Figure 7–8.

Figure 7–8 Assertion Belonging to Two Categories



Assertion Information

Name	Category	Type
Log Message1	security/logging	Logging
WS-Security 1.1 username with certificate	security/authentication, security/msg-pro	wss11-username-with-certificates
Log Message2	security/logging	Logging

- A security policy can contain any number of security_log_template assertions. For example, if you view any of the predefined security policies, you will see two logging assertions included.

Oracle recommends that you create one policy for authentication and message protection, and a second policy for authorization. If you create a policy that contains both an authentication and an authorization assertion, then the authentication assertion must precede the authorization assertion.

When you validate your policies, the validation process checks to see that your policies meet these requirements. If the validation fails during policy creation, the policy is created but is marked as disabled.

To validate a policy:

1. From the Create Policy or Edit Policy page, make any changes to your policy.
2. Click **Validate**.

   If successful, the *Validation successful* message appears.

   If not successful, the resulting error message describes the problem.

7.8 Editing Web Service Policies

You can make changes to the policies you create or to the predefined policies that come with the product. However, Oracle recommends that you do not change the predefined policies so that you will always have a known set of valid policies to work with.

The changes take effect at the next polling interval for policy changes. If you are using a database-based metadata repository, each time you save a change to your policy, a new version is created, and the older versions are retained.

To edit Web service policies:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select a policy from the Policies table and click **Edit**.
3. On the Edit Policy page, make the changes to the policy.
4. Click **Save**.

7.9 Versioning Web Service Policies

Whenever a change to a policy is saved, this results in a new version of the policy being automatically created and the version number being incremented. The Policy Manager maintains the history of these changes, and you can go back to an earlier version.

For example, you might find it useful to create two different versions of a policy, perhaps one with logging and one without, and alternate between them. As another example, you might have an occasional need to use a policy such as *oracle/binding_authorization_denyall_policy* policy with selected roles to temporarily lock down access to a Web service.

By using the versioning feature, you can reuse multiple versions of a policy without having to recreate them every time you need them.

The following sections describe versioning in more detail:

- "Viewing the Version History of Web Services Policies" on page 7-17
- "About the Restore and Activate Policy Options" on page 7-18
- "Creating a New Version of a Web Service Policy" on page 7-19
- "Restoring an Earlier Version of a Web Service Policy" on page 7-19
- "Deleting Versions of a Web Service Policy" on page 7-20

> **Note:** The versioning feature described in this section requires that you use a database-based Oracle WSM Repository. If you are not using a database-based repository, versioning information is not maintained or displayed.

7.9.1 Viewing the Version History of Web Services Policies

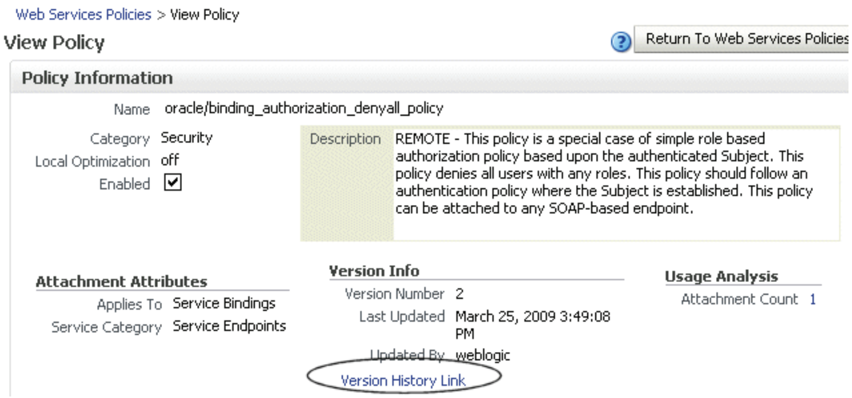
You can view the version history for a Web service policy from the Web Services Policy page, as described in the following procedure.

To view the Web services policy version history:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policy page, select a policy from the Policies table and click **View**.

   In the Policy Information section, you see the version information, including the Version Number of the active version and the date that the policy was last updated.
3. In the View Policy page, click **Version History Link** (Figure 7–9) to go to the View Policy Version History page.

Figure 7–9 Version History Link on the Edit Policy Page



Web Services Policies > View Policy

View Policy

Policy Information

Name oracle/binding_authorization_denyall_policy

Category Security

Local Optimization off

Enabled ☑

Description REMOTE - This policy is a special case of simple role based authorization policy based upon the authenticated Subject. This policy denies all users with any roles. This policy should follow an authentication policy where the Subject is established. This policy can be attached to any SOAP-based endpoint.

Attachment Attributes

Applies To Service Bindings

Service Category Service Endpoints

Version Info

Version Number 2

Last Updated March 25, 2009 3:49:08 PM

Updated By weblogic

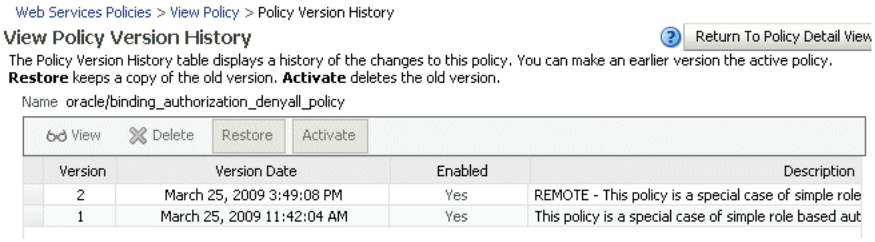
Version History Link

Usage Analysis

Attachment Count 1

4. The policies appear in order in the Policy Version History table with the active policy shown first (Figure 7–10). The active policy has the highest version number, and is the only policy that can be attached to a subject. However, you can make an earlier version of a policy the active policy.

Figure 7–10 View Policy Version History Page



Web Services Policies > View Policy > Policy Version History

View Policy Version History

The Policy Version History table displays a history of the changes to this policy. You can make an earlier version the active policy. **Restore** keeps a copy of the old version. **Activate** deletes the old version.

Name oracle/binding_authorization_denyall_policy

Version	Version Date	Enabled	Description
2	March 25, 2009 3:49:08 PM	Yes	REMOTE - This policy is a special case of simple role
1	March 25, 2009 11:42:04 AM	Yes	This policy is a special case of simple role based aut

7.9.2 About the Restore and Activate Policy Options

You can make an earlier version active by selecting a policy from the Policy Version History table (Figure 7–10), and clicking either the Restore or Activate Policy buttons. In both instances, the selected policy is made the current, active policy, and the policy version number is incremented. The following describes the difference between the Restore and Activate Policy options:

- Clicking **Restore**, the earlier version of the policy is retained. You can make the earlier version the active version without deleting it. Use Restore if you are modifying your policy and want to keep earlier versions of the policy.
- Clicking **Activate Policy**, the selected policy is now the current active policy. The earlier version of the policy is deleted, and the current version is incremented by 1. For example, assume that you have version 1 and version 3 of the policy. You select version 1 and click **Activate Policy**. The policy is activated as version 4, and version 1 is deleted.

  The Activate Policy option can be used in situations where you need to switch between different versions, but you do not want to keep adding policy versions. For example, you may use one version of the policy during business hours and another version during non-business hours. You want to switch between the versions, but you do not want to accumulate multiple versions of the same policy. Therefore, you use Activate Policy to delete the earlier version.

You can also delete any version of the policy, except the active policy, from the Policy Version History table by selecting the policy and clicking **Delete.** You cannot edit the policy from the Policy Version History page. You must edit a policy from the Web Services Management page.

7.9.3 Creating a New Version of a Web Service Policy

You create a new version of an existing Web service policy by making any desired changes and saving the policy.

> **Note:** Save does an implicit validation. If the validation fails, the policy is persisted, but the status is set to **Disabled**.

To create a new version of a Web service policy:

1. From the Edit Policy page, make a change to your policy.
2. Click **Save.**

In the Policy Information section of the page, the version number for the policy is incremented by 1.

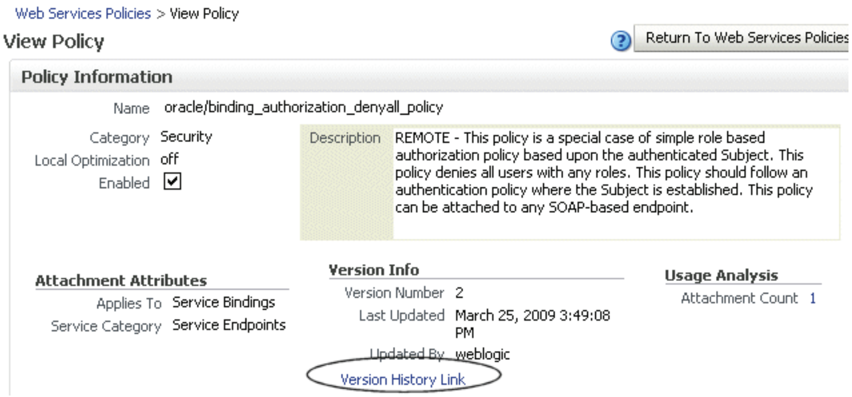
7.9.4 Restoring an Earlier Version of a Web Service Policy

Follow the procedure below to return to an earlier version of a policy.

To restore an earlier version of a Web service policy

1. From the View Policy page, click **Version History Link**, as shown in Figure 7–11.

Figure 7–11 Version History Link on Edit Policy Page



Web Services Policies > View Policy

View Policy

Policy Information

Name oracle/binding_authorization_denyall_policy

Category Security

Local Optimization off

Enabled ☑

Description REMOTE - This policy is a special case of simple role based authorization policy based upon the authenticated Subject. This policy denies all users with any roles. This policy should follow an authentication policy where the Subject is established. This policy can be attached to any SOAP-based endpoint.

Attachment Attributes

Applies To Service Bindings

Service Category Service Endpoints

Version Info

Version Number 2

Last Updated March 25, 2009 3:49:08 PM

Updated By weblogic

Version History Link

Usage Analysis

Attachment Count 1

2. In the Policy History table, select a policy and click **Restore** or click **Activate Policy** .

> **Note:** *Restore* saves the earlier version of the policy, and *Activate Policy* deletes the earlier version.

If you click **Restore**, the selected policy is now the current active policy. The earlier version of the policy is retained, and the current version is incremented by 1.

If you click **Activate Policy**, the selected policy is now the current active policy. The earlier version of the policy is deleted, and the current version is incremented by 1.

7.9.5 Deleting Versions of a Web Service Policy

Follow the procedure below to permanently remove earlier versions of a policy. You can delete all versions except the active policy version. To delete all versions of the policy, including the active version, see "Deleting Web Service Policies" on page 7-20.

To delete a Web service policy version

1. From the Copy Policy page or the Edit Policy Detail page, click **Version History Link**.
2. In the Policy History table, select the policy want to remove, and click **Delete**.
3. A dialog box appears with a message asking you to confirm the deletion. Click **OK**.

The selected policy is deleted from the Oracle WSM Repository and the Policy History table.

7.10 Exporting Web Service Policies

You might want to export a policy to copy it from a development environment to a production environment, or to simply view the policy in another tool or application. Follow the procedure in this section to export a policy from the Oracle WSM Repository. Once the policy is exported, you can import it to another policy store, attach it to Web services, make changes to it, and so forth.

To export a Web service policy

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. Select the policy that you want to export from the list.
3. From the Web Services Policies page, click **Export to File.**
4. Save the policy in the filename of your choice. (Use only ASCII characters in the filename.)

> **Note:** You cannot prefix the name of a policy with oracle_. When you export a predefined policy file, the file is renamed from oracle/<*policyname*> to oracle_<*policyname*>. You should change this name. Otherwise, you will receive exceptions when trying to use the policy.

7.11 Deleting Web Service Policies

Before you delete a policy, Oracle recommends that you verify that the policy is not attached to any policy subjects. You can see the policy subjects that are attached to a policy by doing a policy dependency analysis. See "Analyzing Policy Usage" on page 7-26 for more information. If you try to delete a policy that is attached to a subject, you will receive a warning. You will not be prevented from deleting an attached policy. However, the Web service request will fail the next time the subject to which the policy is attached is invoked.

When you delete a policy, the active policy and all previous versions of the policy are deleted. To retain the active policy version and delete only the previous versions of the policy, see "Versioning Web Service Policies" on page 7-16.

To delete a Web service policy:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select a policy from the Policies table and click **Delete**.
3. A dialog box appears asking you to confirm the deletion. Click **OK**.

7.12 Generating Client Policies

Once you have created the service policy, you can use the Web service WSDL to generate an equivalent client policy with the parameters required to call that service.

You must use the Oracle WSDL instead of the standard WSDL to generate the client policy. The URL for the Web service must be appended with *?orawsdl*, instead of *?wsdl.* Generating the policy increases the likelihood that the client policy will work with the service policy.

Once a policy is generated, you can edit the policy. The policy is populated with the client assertion that is the matching pair to the service assertion. For example, if the service policy contained the assertion, wss_http_token_*service*_template, then the generated client policy is populated with its counterpart, wss_http_token_*client*_template.

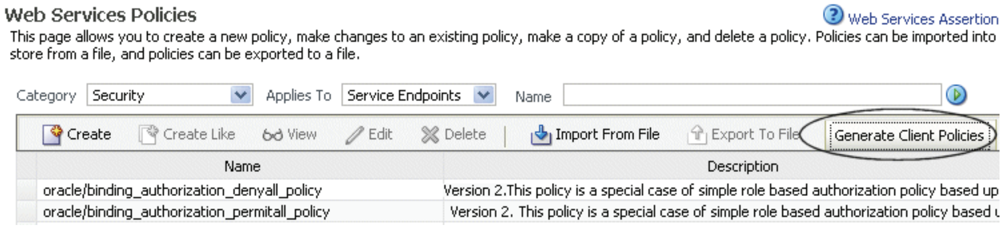
However, the client security policies that are generated will not contain any configuration information. Therefore, once the policies are generated, use the client assertion template and import the configuration information into your client policy. In the example, you would import configuration information from the client assertion template, *wss_http_token_client_template*. After you have made the desired changes to the policy, you must save the policy. Once a policy is saved, you can access it from the Web Services Management page.

You can also delete any generated policies that you do not need. For example, you may want to delete duplicates of already existing MTOM or Reliable Messaging policies.

To generate a Web service client policy

1. Determine the WSDL for the Web service for which you want to generate a Web service client policy.
2. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
3. From the Web Services Policies page, click **Generate Client Policies**, as shown in Figure 7–12.

Figure 7–12 Generate Client Policies on the Web Services Policies Page



Web Services Policies

This page allows you to create a new policy, make changes to an existing policy, make a copy of a policy, and delete a policy. Policies can be imported into store from a file, and policies can be exported to a file.

Category: Security; Applies To: Service Endpoints; Name:

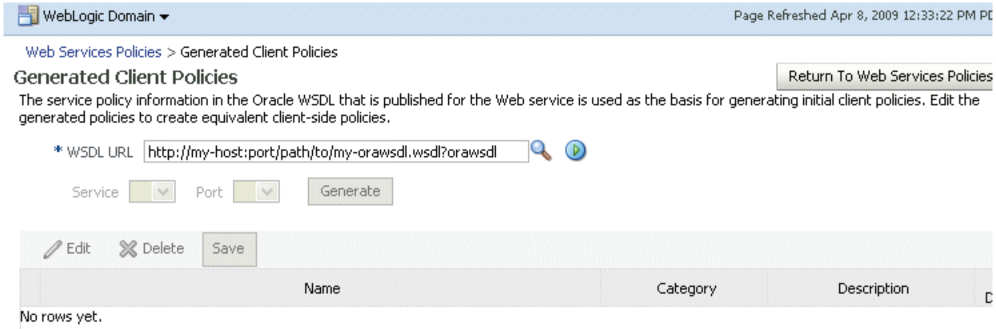
Name	Description
oracle/binding_authorization_denyall_policy	Version 2.This policy is a special case of simple role based authorization policy based up
oracle/binding_authorization_permitall_policy	Version 2. This policy is a special case of simple role based authorization policy based u

4. In the Generated Client Policies page, enter the URL to the Web service WSDL using the following format: *Web_service_endpoint*?orawsdl, and click the control to access the Web service and ports, as shown in Figure 7–13.

> **Note:** You must use *?orawsdl*, instead of *?wsdl*, to get the WSDL that is used to generate the corresponding client policy. Prepend *ora* to *wsdl* to accomplish this.

The *Web_service_endpoint* is the URL to the Web service. The service policy information in the Oracle WSDL published for the Web service is used as the basis for generating the initial client policies.

Figure 7–13 Getting the Web Service and Ports



WebLogic Domain ▾ Page Refreshed Apr 8, 2009 12:33:22 PM PD

Web Services Policies > Generated Client Policies

Generated Client Policies

The service policy information in the Oracle WSDL that is published for the Web service is used as the basis for generating initial client policies. Edit the generated policies to create equivalent client-side policies.

* WSDL URL: http://my-host:port/path/to/my-orawsdl.wsdl?orawsdl

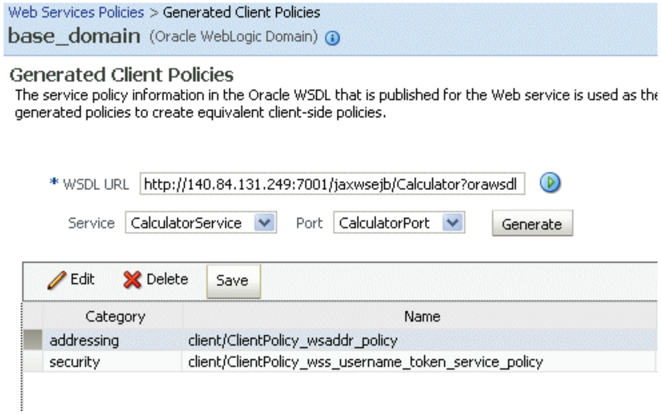
Service: Port:

Name	Category	Description

No rows yet.

5. In the Generated Client Policies page (Figure 7–14), click **Generate** to generate the client policies, as shown in Figure 7–14.

Figure 7–14 Generated Client Policies Page



Web Services Policies > Generated Client Policies

base_domain (Oracle WebLogic Domain)

Generated Client Policies

The service policy information in the Oracle WSDL that is published for the Web service is used as the generated policies to create equivalent client-side policies.

* WSDL URL: http://140.84.131.249:7001/jaxwsejb/Calculator?orawsdl

Service: CalculatorService Port: CalculatorPort

Category	Name
addressing	client/ClientPolicy_wsaddr_policy
security	client/ClientPolicy_wss_username_token_service_policy

6. Select a generated policy from the table and click **Edit**.
7. In the Edit Policy page, edit the policy as necessary.
8. Click **Validate** to validate your changes.
9. Click **Save** to save the changes to your policy.
10. You are returned to the Generated Client Policies page. Edit the other policies as needed.

Once the policy is saved, you can navigate to the Web Services Management page and find the policy in the Policies table.

7.13 Enabling or Disabling a Policy for a Single Policy Subject

When a policy is attached to a Web service, it is enabled by default. You may temporarily disable a policy for a single endpoint without disassociating it from the Web service. When the policy is disabled for an endpoint, it is not enforced for that endpoint.

7.13.1 Using Fusion Middleware Control

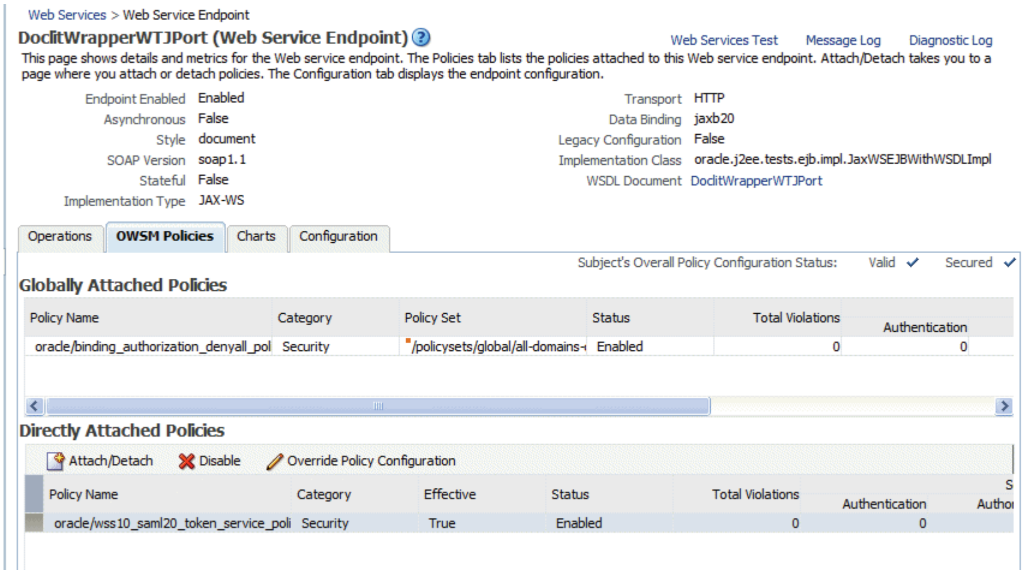
Policies must be individually enabled or disabled for the endpoint; you cannot enable or disable multiple policies at the same time.

To enable or disable a policy attachment:

1. From the Web Service Endpoint page, click the **OWSM Policies** tab.
2. Select the policy you want to enable or disable.

   For Oracle Infrastructure Web services, select a policy from the Directly Attached Policies table.
3. Select **Enable** or **Disable** to enable or disable the policy, respectively, and confirm your selection. (See Figure 7–15.)

Figure 7–15 Enabling or Disabling a Policy Attachment



DoclitWrapperWTJPort (Web Service Endpoint)

This page shows details and metrics for the Web service endpoint. The Policies tab lists the policies attached to this Web service endpoint. Attach/Detach takes you to a page where you attach or detach policies. The Configuration tab displays the endpoint configuration.

Endpoint Enabled	Enabled	Transport	HTTP
Asynchronous	False	Data Binding	jaxb20
Style	document	Legacy Configuration	False
SOAP Version	soap1.1	Implementation Class	oracle.j2ee.tests.ejb.impl.JaxWSEJBWithWSDLImpl
Stateful	False	WSDL Document	DoclitWrapperWTJPort
Implementation Type	JAX-WS		

Subject's Overall Policy Configuration Status: Valid ✓ Secured ✓

Globally Attached Policies

Policy Name	Category	Policy Set	Status	Total Violations	Authentication
oracle/binding_authorization_denyall_pol	Security	/policysets/global/all-domains--	Enabled	0	0

Directly Attached Policies

Policy Name	Category	Effective	Status	Total Violations	Authentication	S Autho
oracle/wss10_saml20_token_service_poli	Security	True	Enabled	0	0	

7.13.2 Using WLST

To enable or disable a policy or multiple policies attached to an endpoint (port):

> **Note:** To enable or disable a client policy using WLST, see "Enabling and Disabling Web Service Client Policies Using WLST" on page 8-18.

1. Connect to the running instance of WebLogic Server for which you want to view the Web services as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Use the `listWebServicePolicies` WLST command to display a list of the Web service policies attached to the desired port.

   ```
   listWebServicePolicies(application,moduleOrCompName,moduleType,serviceName,
   subjectName)
   ```

   For example, to see a list of the policies attached to the `WsdlConcretePort`, use the following command:

   ```
   wls:/base_domain/domainRuntime> listWebServicePolicies('/base_domain/soa_
   server1/jaxwsejb30ws',
   'jaxwsejb','web','{http://namespace/}WsdlConcreteService','WsdlConcretePort')

   WsdlConcretePort :
   security : oracle/binding_authorization_denyall_policy , enabled=true
   security : oracle/wss_username_token_service_policy , enabled=true
   ```

3. Enable or disable a single policy using the `enableWebServicePolicy` command and setting the `enable` argument to `true` or `false`, respectively.

   ```
   enableWebServicePolicy(application, moduleOrCompName, moduleType, serviceName,
   subjectName,
   policyURI,[enable], [subjectType=None] ))
   ```

   For example, to disable the `oracle/binding_authorization_denyall_policy`, enter the following command:

   ```
   wls:/base_domain/domainRuntime> enableWebServicePolicy('/base_domain/soa_
   server1/jaxwsejb30ws',
   'jaxwsejb','web','{http://namespace/}WsdlConcreteService','WsdlConcretePort','o
   racle/binding_authorization_denyall_policy',false)
   ```

4. Enable or disable multiple policies attached to a port using the `enableWebServicePolicies` command and setting the `enable` argument to `true` or `false`, respectively.

   ```
   enableWebServicePolicies(application, moduleOrCompName, moduleType,
   serviceName, subjectName,
   policyURIs,[enable],[subjectType=None] ))
   ```

   For example:

   ```
   wls:/base_domain/domainRuntime> enableWebServicePolicies('/base_domain/soa_
   server1/jaxwsejb30ws',
   'jaxwsejb','web','{http://namespace/}WsdlConcreteService','WsdlConcretePort',
   ['oracle/binding_authorization_denyall_policy',oracle/wss_username_token_
   service_policy],false)
   ```

5. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

For more information about these WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

7.14 Enabling or Disabling a Policy for All Subjects

When a policy is created, it is enabled by default unless it has validation errors. A policy can be globally enabled or disabled from the Edit Policy page. You can enable or disable the policy from one central location, and it will be enabled or disabled for any policy subject to which it is attached.

When you disable a policy from the Edit Policy page, the policy continues to be attached to the policy subjects, but the policy is not enforced. You may want to temporarily disable a policy if you discover that there is a problem with the policy that is causing all requests to a Web service to fail. Once the problem is corrected, you can globally enable the policy.

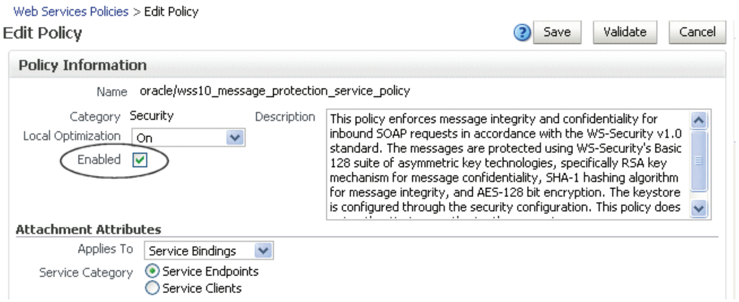
Before disabling a policy, you may want to click **Usage Analysis Link** (see "Analyzing Policy Usage" on page 7-26) to see the policy subjects to which the policy is attached. The change to the policy takes effect at the next polling interval for policy changes.

You may also selectively enable or disable a policy for a specific policy subject rather than for all policy subjects. See "Enabling or Disabling a Policy for a Single Policy Subject" on page 7-23 for more information.

To enable or disable a Web service policy for all policy subjects:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. Select a policy from the Policies table and click **Edit**.
3. In the Policy Information section of the Edit Policy page, select or deselect the **Enabled** box to enable or disable the policy, respectively (see Figure 7–16).

Figure 7–16 Enabled Box on the Edit Policy Page



Web Services Policies > Edit Policy

Edit Policy

Policy Information

Name oracle/wss10_message_protection_service_policy

Category Security

Local Optimization On

Enabled ☑

Description This policy enforces message integrity and confidentiality for inbound SOAP requests in accordance with the WS-Security v1.0 standard. The messages are protected using WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. The keystore is configured through the security configuration. This policy does

Attachment Attributes

Applies To Service Bindings

Service Category ● Service Endpoints ○ Service Clients

4. Click **Save**.

7.15 Enabling or Disabling Assertions Within a Policy

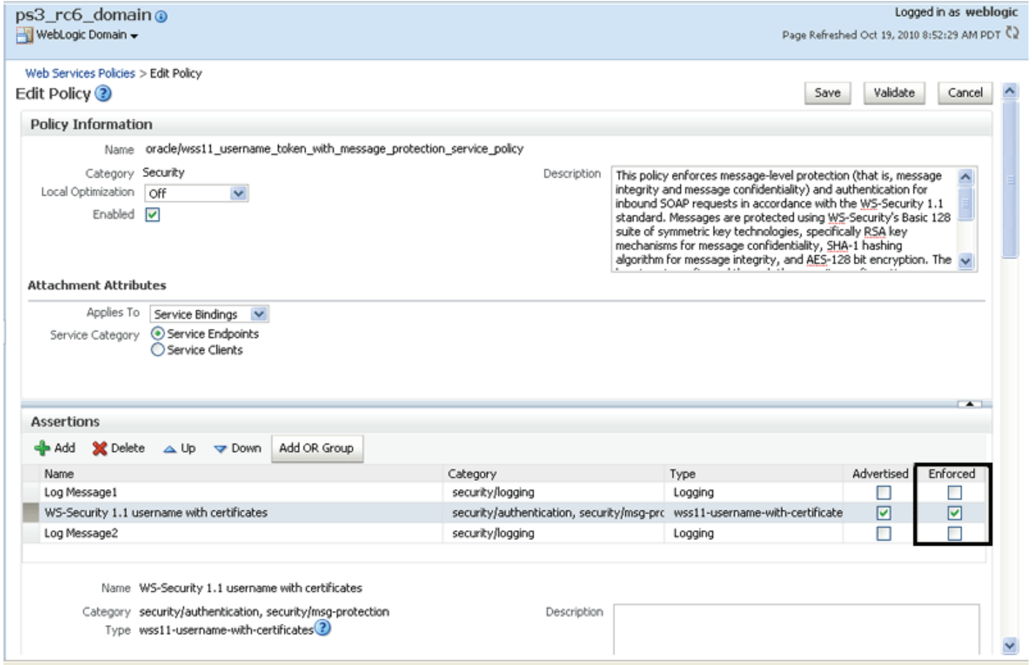
Rather than enable or disable an entire policy, you may wish to enable or disable one or more of the assertions that are contained within a policy. This provides a more fine-grained level of control over the assertions that are executed.

For example, all predefined Web service security policies contain an instance of the logging assertion template, oracle/security_log_template, to capture the entire SOAP message before and after the primary security assertion is executed. By default, the log assertion is not enforced. You must enable it in order for the SOAP message to be logged in message logs. (It is recommended that the logging assertion be enabled for debugging and auditing purposes only. For more information about logging, see "Diagnosing Problems Using Logs" on page 16-20.)

To enable or disable one or more assertions within a policy:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. Select a policy from the Policies table and click **Edit**.
3. In the Assertions section of the **Edit Policy** page, select or deselect the **Enforced** box to enable or disable the assertion within the policy, respectively (see Figure 7–17).

Figure 7–17 Enable or Disable an Assertion Within a Policy



ps3_rc6_domain
WebLogic Domain

Logged in as weblogic
Page Refreshed Oct 19, 2010 8:52:29 AM PDT

Web Services Policies > Edit Policy

Edit Policy

Policy Information

Name oracle/wss11_username_token_with_message_protection_service_policy

Category Security

Local Optimization Off

Enabled ☑

Description: This policy enforces message-level protection (that is, message integrity and message confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. Messages are protected using WS-Security's Basic 128 suite of symmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. The

Attachment Attributes

Applies To Service Bindings

Service Category ● Service Endpoints ○ Service Clients

Assertions

Name	Category	Type	Advertised	Enforced
Log Message1	security/logging	Logging	□	□
WS-Security 1.1 username with certificates	security/authentication, security/msg-prc	wss11-username-with-certificate	☑	☑
Log Message2	security/logging	Logging	□	□

Name WS-Security 1.1 username with certificates

Category security/authentication, security/msg-protection

Type wss11-username-with-certificates

Description

4. Click **Save**.

7.16 Analyzing Policy Usage

Note: The policy usage feature described in this section requires that you use a database-based Oracle WSM Repository. If you are not using a database-based repository, policy usage information is not available.

Policies are created and managed at the domain level. The central management of policies gives you the ability to reuse policies and attach them to multiple policy subjects. Any change to a policy (for example, editing a policy or deleting a policy) affects all policy subjects to which the policy is attached. Therefore, before making any changes to your policies, Oracle recommends you do a usage analysis to see which subjects are using a particular policy.

> **Note:** The usage analysis simply identifies which policy subjects will be affected; it does not define the effect of the change. You need to evaluate the change on each of the policy subjects and determine if you should proceed.

To perform a usage analysis:

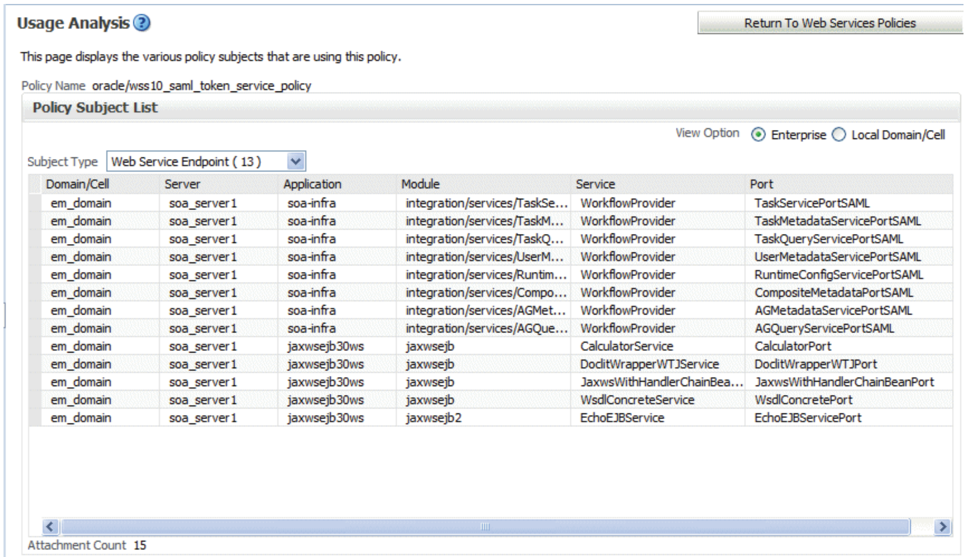
1. Navigate to the Web Services Policies page as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.

   The Attachment Count column of the Policies table shows the number of subjects to which a policy is attached.

2. Click the number in the Attachment Count column for the selected policy to display the Usage Analysis page (Figure 7–18).

   Alternatively, you can select the policy from the Policies table and click **View**. In the Policy Information region of the page, click the **Attachment Count** number in the **Usage Analysis** field to display the Usage Analysis page.

Figure 7–18 Usage Analysis for a Policy



Usage Analysis

This page displays the various policy subjects that are using this policy.

Policy Name oracle/wss10_saml_token_service_policy

Policy Subject List

View Option ● Enterprise ○ Local Domain/Cell

Subject Type Web Service Endpoint (13)

Domain/Cell	Server	Application	Module	Service	Port
em_domain	soa_server1	soa-infra	integration/services/TaskSe...	WorkflowProvider	TaskServicePortSAML
em_domain	soa_server1	soa-infra	integration/services/TaskM...	WorkflowProvider	TaskMetadataServicePortSAML
em_domain	soa_server1	soa-infra	integration/services/TaskQ...	WorkflowProvider	TaskQueryServicePortSAML
em_domain	soa_server1	soa-infra	integration/services/UserM...	WorkflowProvider	UserMetadataServicePortSAML
em_domain	soa_server1	soa-infra	integration/services/Runtim...	WorkflowProvider	RuntimeConfigServicePortSAML
em_domain	soa_server1	soa-infra	integration/services/Compo...	WorkflowProvider	CompositeMetadataPortSAML
em_domain	soa_server1	soa-infra	integration/services/AGMet...	WorkflowProvider	AGMetadataServicePortSAML
em_domain	soa_server1	soa-infra	integration/services/AGQue...	WorkflowProvider	AGQueryServicePortSAML
em_domain	soa_server1	jaxwsejb30ws	jaxwsejb	CalculatorService	CalculatorPort
em_domain	soa_server1	jaxwsejb30ws	jaxwsejb	DoclitWrapperWTJService	DoclitWrapperWTJPort
em_domain	soa_server1	jaxwsejb30ws	jaxwsejb	JaxwsWithHandlerChainBea...	JaxwsWithHandlerChainBeanPort
em_domain	soa_server1	jaxwsejb30ws	jaxwsejb	WsdlConcreteService	WsdlConcretePort
em_domain	soa_server1	jaxwsejb30ws	jaxwsejb2	EchoEJBService	EchoEJBServicePort

Attachment Count 15

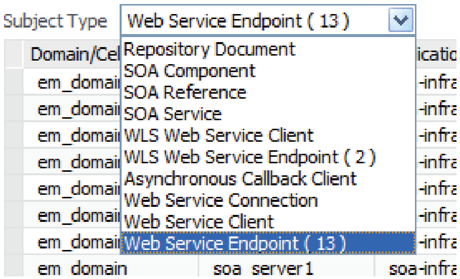
   The Policy Subject List is filtered by subject type. The table displays a list of the policy subjects, of the selected type, to which the policy is attached. Valid subject types include Oracle WSM Repository Documents, WLS Web Service Client, WLS Web Service Endpoint, and the subject types listed in "Resource Type" on page 9-18. Note that the Policy Subject List summary table displays fields that are relevant to the selected policy subject type only.

   The total number of policy subjects to which the policy is attached is shown at the bottom of the page in the **Attachment Count** field.

3. To view the other policy subjects to which the policy is attached, select the subject type from the **Subject Type** menu.

The **Subject Type** menu provides an attachment count for each subject type to which the policy is attached.

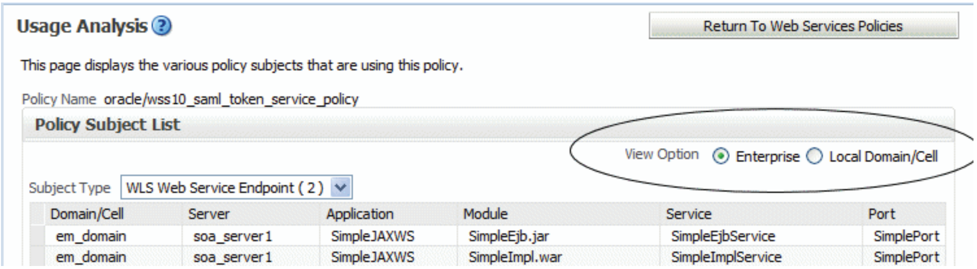
Figure 7–19 Subject Type Menu on Usage Analysis Page



Subject Type: Web Service Endpoint (13)

4. In cases where multiple domains share the same Oracle WSM Repository to store Oracle WSM metadata, you can specify whether you want to view policy subjects in the Local Domain or in all domains in the enterprise. To view the policy subjects for all domains in the enterprise, select Enterprise in the **View Option** field.

Figure 7–20 View Option Field on Usage Analysis Page



This page displays the various policy subjects that are using this policy.

Policy Name oracle/wss10_saml_token_service_policy

Policy Subject List

View Option ● Enterprise ○ Local Domain/Cell

Subject Type: WLS Web Service Endpoint (2)

Domain/Cell	Server	Application	Module	Service	Port
em_domain	soa_server1	SimpleJAXWS	SimpleEjb.jar	SimpleEjbService	SimplePort
em_domain	soa_server1	SimpleJAXWS	SimpleImpl.war	SimpleImplService	SimplePort

Please note:

- Both enabled and disabled policy references are included in the policy usage count. For information about disabling a policy reference, see "Enabling or Disabling a Policy for a Single Policy Subject" on page 7-23 and "Enabling or Disabling a Policy for All Subjects" on page 7-25.
- After attaching a policy to an Oracle Infrastructure Web Service endpoint, you need to restart the Web service application to display an accurate policy usage count. You do not need to restart a SOA composite or a WebLogic Java EE Web service application.
- You must invoke an ADF DC client to display an accurate policy usage count.

7.17 Policy Advertisement

For a standard WSDL (?wsdl), you can publish different version combinations for WS-Policy and WS-SecurityPolicy. For example, http://localhost:8080/abc?wsdl&wsp=1.5&wssp=1.2 returns a WSDL with the following policy versions published: WS-Policy 1.5 and WS-SecurityPolicy 1.2.

Note: For an Oracle WSDL (?orawsdl), you cannot advertise different version combinations for WS-Policy and WS-SecurityPolicy. For ?orawsdl, the policy is advertised with the following versions only: WS-Policy 1.2 and WS-SecurityPolicy 1.1 with Oracle extensions.

Table 7–2 lists the valid version combinations.

Table 7–2 Policy Advertisement

Version Combination	Description
?wsdl	WS-Policy 1.2 and WS-SecurityPolicy 1.1
?wsdl&wsp=1.5	WS-Policy version 1.5 and WS-SecurityPolicy 1.3
?wsdl&wssp=1.2	WS-Policy versions 1.5 and WS-SecurityPolicy 1.2
?wsdl&wssp=1.3	WS-Policy versions 1.5 and WS-SecurityPolicy 1.3
?wsdl&wsp=1.5&wssp=1.2	WS-Policy 1.5 and WS-SecurityPolicy 1.2
?wsdl&wsp=1.5&wssp=1.3	WS-Policy 1.5 and WS-SecurityPolicy 1.3
?wsdl&wsp=1.2&wssp=1.2	WS-Policy 1.2 and WS-SecurityPolicy 1.2

8 Attaching Policies to Web Services

This chapter includes the following sections:

8.1 Viewing the Policies That are Attached to a Web Service

The following sections describe how to view the policies that are attached to a Web service using Fusion Middleware Control and the WebLogic Scripting Tool (WLST).

8.1.1 Using Fusion Middleware Control

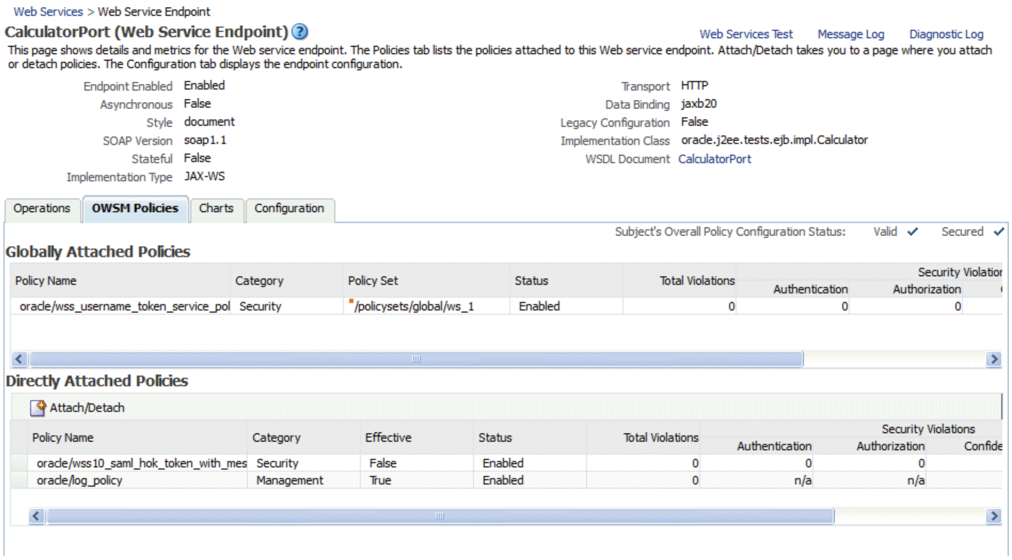
To view the policies that are attached to a Web service:

1. Navigate to the home page for the Web service, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service endpoints if they are not already displayed.
3. Click the name of a endpoint to navigate to the Web Service Endpoints page for a particular Web service.
4. Click the **OWSM Policies** tab.

   Figure 8–1 shows the screen display for an Oracle Infrastructure Web service endpoint that has both globally and directly attached policies. The output displays the globally attached policies that are in effect for the endpoint, all directly attached policies, and whether the endpoint has a valid configuration and is secure. You can view the run-time constraints, if any, configured for a policy set by placing your mouse over the policy set name, or clicking the red dot at the front of the policy set name. For more information about run-time constraints, see "Specifying Run-time Constraints in Policy Sets" on page 9-26.

Because you can specify the priority of a globally or directly attached policy, as described in "Specifying the Priority of a Policy Attachment" on page 9-35, the Effective field for a directly attached policy indicates if it is in effect for the endpoint. Note that to simplify endpoint management, all directly attached policies are shown in the output regardless of whether they are in effect. In contrast, only globally attached policies that are in effect for the endpoint are displayed. For details about effective policies for an endpoint, see "How the Effective Set of Policies is Calculated" on page 9-38.

Figure 8–1 Policies Attached to an Oracle Infrastructure Web Service Endpoint



Web Services > Web Service Endpoint

CalculatorPort (Web Service Endpoint)

This page shows details and metrics for the Web service endpoint. The Policies tab lists the policies attached to this Web service endpoint. Attach/Detach takes you to a page where you attach or detach policies. The Configuration tab displays the endpoint configuration.

Endpoint Enabled: Enabled
Asynchronous: False
Style: document
SOAP Version: soap1.1
Stateful: False
Implementation Type: JAX-WS

Transport: HTTP
Data Binding: jaxb20
Legacy Configuration: False
Implementation Class: oracle.j2ee.tests.ejb.impl.Calculator
WSDL Document: CalculatorPort

Subject's Overall Policy Configuration Status: Valid ✓ Secured ✓

Globally Attached Policies

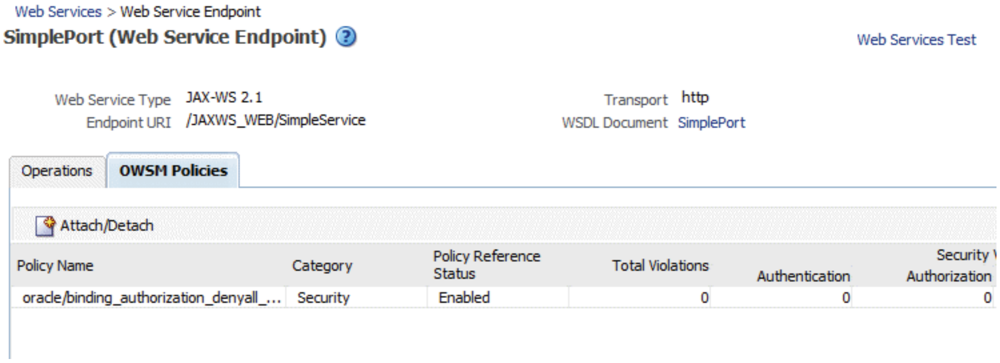
Policy Name	Category	Policy Set	Status	Total Violations	Security Violatior	
					Authentication	Authorization
oracle/wss_username_token_service_pol	Security	/policysets/global/ws_1	Enabled	0	0	0

Directly Attached Policies

Policy Name	Category	Effective	Status	Total Violations	Security Violations		
					Authentication	Authorization	Confide
oracle/wss10_saml_hok_token_with_mes	Security	False	Enabled	0	0	0	
oracle/log_policy	Management	True	Enabled	0	n/a	n/a	

Figure 8–2 shows the screen display for a WebLogic Java EE endpoint. Only policies that are directly attached to an endpoint are displayed. Globally attached policies are not available.

Figure 8–2 Policies Attached to a WebLogic Java EE Web Service Endpoint



Web Services > Web Service Endpoint

SimplePort (Web Service Endpoint)

Web Service Type: JAX-WS 2.1
Endpoint URI: /JAXWS_WEB/SimpleService

Transport: http
WSDL Document: SimplePort

Policy Name	Category	Policy Reference Status	Total Violations	Security \	
				Authentication	Authorization
oracle/binding_authorization_denyall_...	Security	Enabled	0	0	0

8.1.2 Using WLST

Use the following procedure to view the policies that are attached to a Web service:

1. Connect to the running instance of WebLogic Server to which the application is deployed as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Use the listWebServices WLST command to display a list of the Web services in your application as described in "Viewing the Web Services in Your Application" on page 6-5.
3. Use the listWebServicePorts command to display the port name and endpoint URL for a Web service.

   ```
   listWebServicePorts(application,moduleOrCompName,moduleType,serviceName)
   ```

   For example, to display the port for the WsdlConcreteService:

   ```
   wls:/wls-domain/serverConfig>
   listWebServicePorts("/wls-domain/AdminServer/jaxwsejb30ws",
   "jaxwsejb","web","WsdlConcreteService")

   WsdlConcretePort   http://host.example.com:7001/jaxwsejb/WsdlAbstract
   ```

4. Use the listWebServicePolicies command to view the policies that are attached to a Web service port.

   ```
   listWebServicePolicies(application,moduleOrCompName,moduleType,serviceName,subj
   ectName)
   ```

   For example, to view the policies attached to the WsdlConcretePort port and any policy override settings:

   ```
   wls:/wls_domain/serverConfig> listWebServicePolicies("/wls_
   domain/AdminServer/jaxwsejb30ws",
   "jaxwsejb","web","WsdlConcreteService","WsdlConcretePort")

   WsdlConcretePort :
   addressing : oracle/wsaddr_policy , enabled=true
   management : oracle/log_policy , enabled=true
   security : oracle/wss_username_token_service_policy, enabled=true
   Attached policy or policies are valid; endpoint is secure.
   ```

8.2 Attaching Policies to Web Services

The following sections describe how to attach policies to a single subject, to multiple subjects (bulk attachment), and to validate the subject once policies are attached:

- "Attaching a Policy to a Single Subject" on page 8-3
- "Attaching a Policy to Multiple Subjects (Bulk Attachment)" on page 8-8

Note: For WebLogic Java EE Web services policy attachment:

- Only Oracle WSM security policies can be attached.
- Oracle WSM policies and WebLogic Web Service policies cannot be attached to the same endpoint. If a WebLogic Java EE endpoint has WebLogic policies attached, you cannot attach Oracle WSM security policies. Note that WebLogic policies can be attached using the WebLogic Server Administration Console.

8.2.1 Attaching a Policy to a Single Subject

A **subject** is an entity to which a policy can be associated. You can attach one or more policies to a subject.

The order in which policies are attached to a subject or appear in the list of attached policies does not determine the order in which policies are executed. As a message is passed between the client and the Web service, the order of the interceptors in the policy interceptor chain determines the order in which the policies are executed.

See "How Policies are Executed" on page 3-7 for more information.

> **Note:** Policy attachment is not synchronized automatically for SOA, ADF, and WebCenter services in a cluster. When using SOA, ADF, and WebCenter services in a cluster, you must attach and/or detach policies to each instance of the cluster. This issue does not apply to WebLogic Java EE Web services and SOA composite services.

8.2.1.1 Attaching a Policy to a Web Service Using Fusion Middleware Control

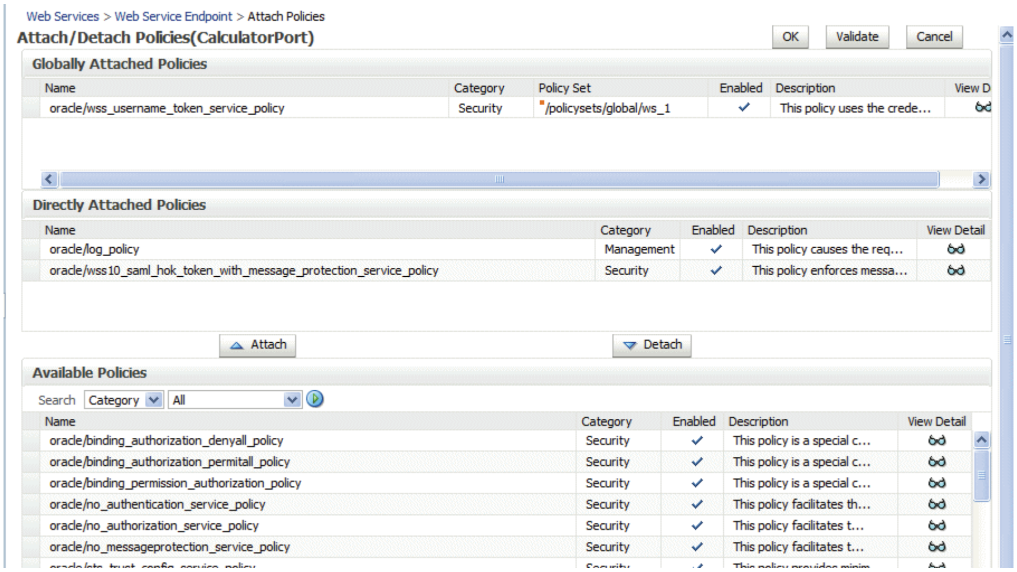
Follow this procedure to attach a policy to a single Web service endpoint. See "Attaching a Policy to Multiple Subjects (Bulk Attachment)" to attach a policy to multiple Web services at the same time.

To attach a policy to a Web service:

1. Navigate to the home page for the Web service, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service endpoints, if they are not already displayed.
3. Click the name of a endpoint to navigate to the Web Service Endpoints page for a particular Web service.
4. Click the **OWSM Policies** tab.

   The policies that are already globally and directly attached to the endpoint are displayed as shown in Figure 8–1.
5. Click **Attach/Detach**.
6. Select a policy from the Available Policies list, and click **Attach**. See Figure 8–3.

Figure 8–3 Attaching Policies to a Web Service



Attach/Detach Policies(CalculatorPort)

Globally Attached Policies

Name	Category	Policy Set	Enabled	Description	View D
oracle/wss_username_token_service_policy	Security	/policysets/global/ws_1	✓	This policy uses the crede...	👓

Directly Attached Policies

Name	Category	Enabled	Description	View Detail
oracle/log_policy	Management	✓	This policy causes the req...	👓
oracle/wss10_saml_hok_token_with_message_protection_service_policy	Security	✓	This policy enforces messa...	👓

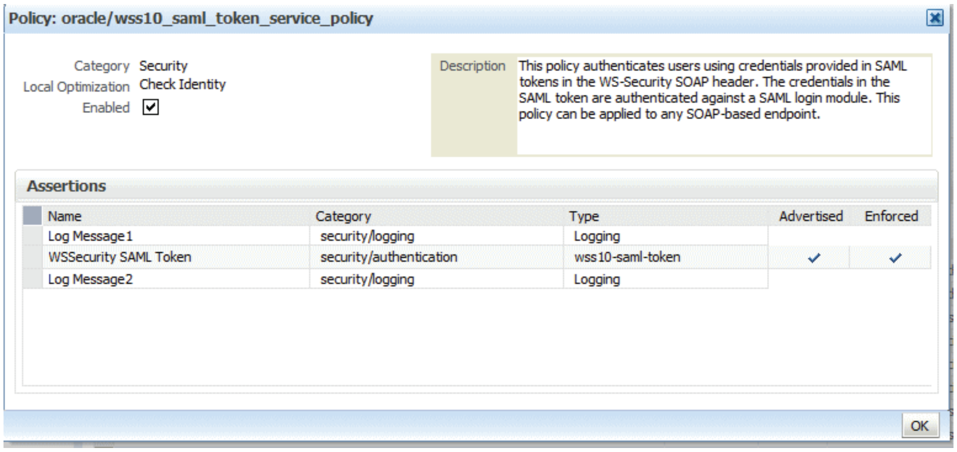
Available Policies

Search Category All

Name	Category	Enabled	Description	View Detail
oracle/binding_authorization_denyall_policy	Security	✓	This policy is a special c...	👓
oracle/binding_authorization_permitall_policy	Security	✓	This policy is a special c...	👓
oracle/binding_permission_authorization_policy	Security	✓	This policy is a special c...	👓
oracle/no_authentication_service_policy	Security	✓	This policy facilitates th...	👓
oracle/no_authorization_service_policy	Security	✓	This policy facilitates t...	👓
oracle/no_messageprotection_service_policy	Security	✓	This policy facilitates t...	👓
oracle/sts_trust_config_service_policy	Security	✓	This policy provides minim	👓

7. To view details about a policy, select the policy and click the **View Detail** icon. A pop-up window provides a full read-only description of the policy and lists the assertions that it contains. See Figure 8–4. Click **OK** when you are finished reviewing the details of the policy.

Figure 8–4 Viewing Details about a Policy



Policy: oracle/wss10_saml_token_service_policy

Category Security
Local Optimization Check Identity
Enabled ☑

Description This policy authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. This policy can be applied to any SOAP-based endpoint.

Assertions

Name	Category	Type	Advertised	Enforced
Log Message1	security/logging	Logging		
WSSecurity SAML Token	security/authentication	wss10-saml-token	✓	✓
Log Message2	security/logging	Logging		

8. Continue selecting and attaching policies. When you are finished, click **Validate** to verify that the combination of policies selected is valid.

9. Click **OK**.

10. The Web Service Endpoint page now displays the attached policy on the **OWSM Policies** tab as shown in Figure 8–1.

> **Note:** The output displays the globally attached policies that are in effect for the endpoint, all directly attached policies, and whether the endpoint has a valid configuration and is secure. Because you can specify the priority of a globally or directly attached policy, as described in "Specifying the Priority of a Policy Attachment" on page 9-35, the Effective field for a directly attached policy indicates if it is in effect for the endpoint. Note that to simplify endpoint management, all directly attached policies are shown in the output regardless of whether they are in effect. In contrast, only globally attached policies that are in effect for the endpoint are displayed. For details about effective policies for an endpoint, see "How the Effective Set of Policies is Calculated" on page 9-38.

11. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite or a WebLogic Java EE Web service application.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

8.2.1.2 Attaching a Policy to a Web Service Using WLST

Use the following procedure to attach (or detach) a single policy, or multiple policies, to a single Web service port using WLST.

1. View the list of policies currently attached to the port as described in "Using WLST" in "Viewing the Policies That are Attached to a Web Service" on page 8-1.
2. View the list of available policies as described in "Displaying a List of the Available Policies Using WLST" on page 7-2.
3. To attach policies, do one of the following:
   - Use the `attachWebServicePolicy` command to attach a single policy to a Web service port. Specify the policy to be attached using the `policyURI` argument. If you specify a policy that is already attached or exists, then this command enables the policy if it is disabled.

     ```
     attachWebServicePolicy(application, moduleOrCompName, moduleType,
     serviceName,
     subjectName, policyURI, [subjectType=None]
     ```

     For example, to attach the policy `oracle/wss_username_token_service_policy` to the `WsdlConcretePort` of the `WsdlConcreteService`, use the following command:

     ```
     wls:/wls_domain/serverConfig> attachWebServicePolicy("/wls_
     domain/AdminServer/jaxwsejb30ws",
     "jaxwsejb","web","WsdlConcreteService","WsdlConcretePort",
     "oracle/wss_username_token_service_policy")
     ```

   - Use the `attachWebServicePolicies` command to attach multiple policies to a Web service port. Specify the policies to be attached using the `policyURIs` argument. If any of the policies that you specify in this command are already attached, then this command enables the policies that are already attached (if they are disabled), and attaches the others.

     ```
     attachWebServicePolicies(application, moduleOrCompName, moduleType,
      serviceName, subjectName, policyURIs, [subjectType=None]
     ```

     For example, to attach the policies `oracle/wss_username_token_service_policy` and `oracle/wsrm10_policy`to the `WsdlConcretePort` of the `WsdlConcreteService`, use the following command:

     ```
     wls:/wls_domain/serverConfig> attachWebServicePolicies("/wls_
     domain/AdminServer/jaxwsejb30ws",
     "jaxwsejb","web","WsdlConcreteService","WsdlConcretePort",
     ["oracle/wss_username_token_service_policy","oracle/wsrm10_policy"])

     Please restart application to uptake the policy changes.
     ```

Note: The policyURIs are validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. If the policy validation fails, a message is displayed and the command is not executed.

If the wsm-pm application is not installed or is not available, these commands are not executed.

For additional information about validating policies, see "Validating Policy Subjects" on page 8-10.

---

4. To detach policies, do one of the following:

   - Use the `detachWebServicePolicy` command to detach a single policy from a Web service port. Specify the policy to be detached using the `policyURI` argument.

     ```
     detachWebServicePolicy(application, moduleOrCompName, moduleType,
      serviceName, subjectName, policyURI, [subjectType=None]
     ```

     For example, to detach the policy `oracle/wss_username_token_service_policy` from the `WsdlConcretePort` of the `WsdlConcreteService`, use the following command:

     ```
     wls:/wls_domain/serverConfig> detachWebServicePolicy("/wls_
     domain/AdminServer/jaxwsejb30ws",
     "jaxwsejb","web","WsdlConcreteService","WsdlConcretePort",
     "oracle/wss_username_token_service_policy")
     ```

   - Use the `detachWebServicePolicies` command to detach multiple policies from a Web service port. Specify the policies to be detached using the `policyURIs` argument.

     ```
     detachWebServicePolicies(application, moduleOrCompName, moduleType,
     serviceName, subjectName, policyURIs, [subjectType=None]
     ```

     For example, to detach the policies `oracle/wss_username_token_service_policy` and `oracle/wsrm10_policy`to the `WsdlConcretePort` of the `WsdlConcreteService`, use the following command:

     ```
     wls:/wls_domain/serverConfig> detachWebServicePolicies("/wls_
     domain/AdminServer/jaxwsejb30ws",
     "jaxwsejb","web","WsdlConcreteService","WsdlConcretePort",
     ["oracle/wss_username_token_service_policy","oracle/wsrm10_policy"])

     Please restart application to uptake the policy changes.
     ```

5. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

8.2.2 Attaching a Policy to Multiple Subjects (Bulk Attachment)

From the Application pages, you can attach one or more policies to one or more Web services.

> **Notes:** The bulk attachment mechanism does not perform validation on the policies that you attach.
>
> The bulk attachment mechanism does not prevent you from creating an unsupported configuration such as having multiple authentication policies, or from attaching the same policy multiple times, and so forth.
>
> Policy attachment is not synchronized automatically for SOA, ADF, and WebCenter services in a cluster. When using SOA, ADF, and WebCenter services in a cluster, you must attach and/or detach policies to each instance of the cluster. This issue does not apply to WebLogic Java EE Web services and SOA composite services.

To attach a policy to multiple Web services within an application:

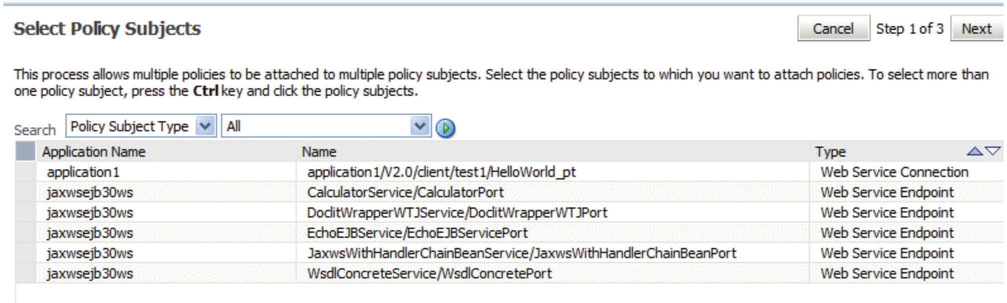
1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to attach the policy.
2. Select the domain, and then the instance of the server to which you want to attach the policy. The server can be an Administration Server or a Managed Server.
3. Using Fusion Middleware Control, click **WebLogic Server** and then **Web Services**.
4. From the Web Services Summary page, click **Attach Policies**.
5. From the Select Policy Subjects page, select one or more applications to which to attach a policy, as shown in Figure 8–5.

   Use the **Search** control to search for a particular policy subject type, a particular application name, or the type of Web service to which you want to attach a policy. Valid policy subject types include: Web Service Endpoint, Web Service Client, Web Service Connection, SOA Component, SOA Service, SOA Reference, Asynchronous Callback Client, or WLS Web Service Endpoint. For more information about asynchronous callback clients, see "Developing Asynchronous Web Services" in *Developer's Guide for Oracle Infrastructure Web Services*.

   For example, if you choose to search for a policy subject type of Web Service Client, only available Web service clients, if any, are displayed.

   To select more than one application, press the Ctrl key and click the applications.

Figure 8–5 Select Subjects Page



Select Policy Subjects

This process allows multiple policies to be attached to multiple policy subjects. Select the policy subjects to which you want to attach policies. To select more than one policy subject, press the **Ctrl** key and click the policy subjects.

Search: Policy Subject Type | All

Application Name	Name	Type
application1	application1/V2.0/client/test1/HelloWorld_pt	Web Service Connection
jaxwsejb30ws	CalculatorService/CalculatorPort	Web Service Endpoint
jaxwsejb30ws	DoclitWrapperWTJService/DoclitWrapperWTJPort	Web Service Endpoint
jaxwsejb30ws	EchoEJBService/EchoEJBServicePort	Web Service Endpoint
jaxwsejb30ws	JaxwsWithHandlerChainBeanService/JaxwsWithHandlerChainBeanPort	Web Service Endpoint
jaxwsejb30ws	WsdlConcreteService/WsdlConcretePort	Web Service Endpoint

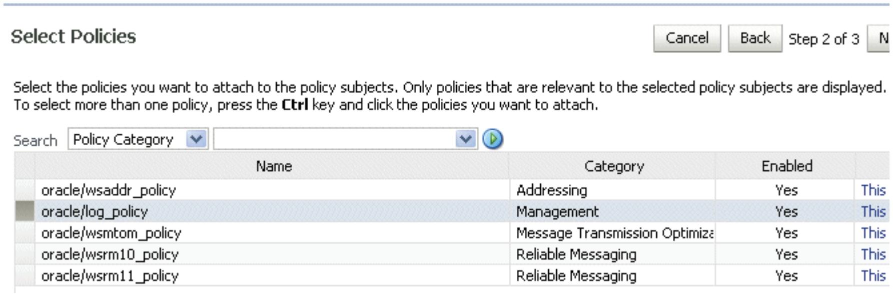
6. Click **Next.**

7. From the Select Policies page, select one or more policies that you want to attach to the selected applications, as shown in Figure 8–6. The Select Policies page shows only those policies that you can apply to all of the subjects selected in the previous step.

   > **Note:** You can attach only security policies to WebLogic Java EE Web service endpoints using Fusion Middleware Control. If you attempt to attach a non-security policy to a WebLogic Web service endpoint, the action is ignored.

   To select more than one policy, press the Ctrl key and click the policies you want to attach.

Figure 8–6 Select Policies Page



Select Policies

Select the policies you want to attach to the policy subjects. Only policies that are relevant to the selected policy subjects are displayed. To select more than one policy, press the **Ctrl** key and click the policies you want to attach.

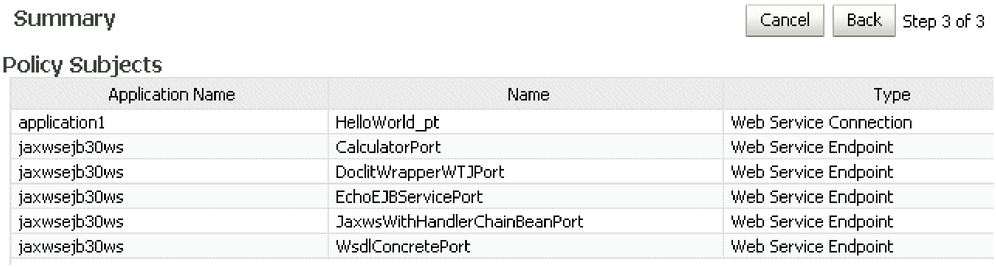
Search: Policy Category

Name	Category	Enabled	
oracle/wsaddr_policy	Addressing	Yes	This
oracle/log_policy	Management	Yes	This
oracle/wsmtom_policy	Message Transmission Optimiz	Yes	This
oracle/wsrm10_policy	Reliable Messaging	Yes	This
oracle/wsrm11_policy	Reliable Messaging	Yes	This

8. Click **Next.**

   The Summary page displays the applications you selected and the policies that will be attached to those applications, as shown in Figure 8–7.

Figure 8–7 Attachment Summary Page



Summary

Policy Subjects

Application Name	Name	Type
application1	HelloWorld_pt	Web Service Connection
jaxwsejb30ws	CalculatorPort	Web Service Endpoint
jaxwsejb30ws	DoclitWrapperWTJPort	Web Service Endpoint
jaxwsejb30ws	EchoEJBServicePort	Web Service Endpoint
jaxwsejb30ws	JaxwsWithHandlerChainBeanPort	Web Service Endpoint
jaxwsejb30ws	WsdlConcretePort	Web Service Endpoint

9. Click **Back** to make any changes, or click **Attach** to complete the bulk attachment.
10. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite or a WebLogic Java EE Web service application.

> **Note:** You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

8.3 Validating Policy Subjects

The type and number of assertions within a policy may be valid and, therefore, a policy may be internally consistent and valid. However, when more than one policy is attached to a policy subject, the combination of policies must also be valid. Specifically, the following must be true:

> **Note:** When you view a policy, only the major category, such as security, is displayed. To see the subtype (such as authorization), see the **Assertion Details** section of the assertion template on which the policy is based.

- Only one MTOM policy can be attached to a policy subject.
- Only one Reliable Messaging policy can be attached to a policy subject.
- Only one WS-Addressing policy can be attached to a policy subject.
- Only one Security policy with subtype authentication can be attached to a subject.
- Only one Security policy with subtype sts-config can be attached to a subject.
- If an authentication policy and an authorization policy are both attached to a policy subject, the authentication policy must precede the authorization policy.
- There may be one or more security policies attached to a policy subject. For example, a security policy can contain an assertion that belongs to the authentication or message protection subtype categories, or an assertion that belongs to both subtype categories. The second security policy contains an assertion that belongs to the authorization subtype.
- If the policies attached to a subject are exact duplicates of each other, including any configuration overrides, the policy attachment is viewed as a duplicate and the configuration is valid.
- If the policy requires a particular transport protocol (for example, HTTP or HTTPS), it checks to see that the Web service uses the expected transport protocol. (The check is done at run time.)

The run time automatically enforce STS-Trust configuration policies first and authorization policies last

You cannot use policy subject validation to check the validity of multiple policy subjects when you use the bulk attachment feature. After you attach the policies to your subjects with this feature, you must validate each subject individually.

> **Note:** The policy subject validation does not validate the XML schema of the policy. Therefore, if you manually edit the policy file, you must use another tool to check that the XML is valid.

To check for policy subject validation:

1. From the navigator pane, click the plus sign (**+**) for the Application Deployments folder to expose the applications in the farm, and select the application.

   The Application Deployment home page is displayed.

2. Using Fusion Middleware Control, click **Application Deployment**, then click **Web Services**.

   This takes you to the Web Services summary page for your application.

3. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
4. Click the name of the port to navigate to the Web Service Endpoints page.
5. Click the **Policies** tab.
6. Click **Attach/Detach.**
7. Click **Validate.**

   If there is a validation error, a dialog box appears describing the error. Fix the error and do a policy subject validation again.

8.4 Attaching Policies to Web Service Clients

This section describes how to attach policies to SOA references, connection-based Web service clients (such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client), asynchronous Web service Callback clients and Java EE Web service clients.

> **Note:** For information about programmatically attaching Oracle WSM policies to RESTful Web service clients at design time, see "Securing RESTful Web Service Clients" in *Using the Jersey JAX-RS Reference Implementation*.

When using WLST to attach policies to a Web service client, the steps that you follow are the same for all Web service client types. The argument settings specify the type of client to which you are attaching the policy. For more information, see "Attaching Policies to Web Service Clients Using WLST" on page 8-15.

8.4.1 Attaching Policies to Web Service Clients Using Fusion Middleware Control

In Fusion Middleware Control, the steps you follow to attach a policy to a Web service client are the same for all Web service client types. However, how you navigate to the Web service client varies based on the application type, as described in the following sections.

8.4.1.1 Attaching Policies to SOA References

The following procedures describe how to attach policies to SOA references. For more information about developing SOA references, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

To attach policies to a SOA reference:

1. View the SOA reference, as described in "Viewing SOA References" on page 6-10.
2. Select the **Policies** tab.
3. In the Directly Attached Policies section of the page, click **Attach/Detach**.
4. From the Available Policies section of the page, select one or more policies that you want to attach. Click **Validate** to validate the policy, or **Check Services Compatibility** to make sure that the client policies are compatible with the service policies.
5. Click **Attach** when you are sure that you want to attach the policy or policies.
6. Click **OK**.

8.4.1.2 Attaching Policies to Connection-Based Web Service Clients

The following procedure describes how to attach policies to a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.

For more information about developing ADF DC Web service clients, see "Using ADF Model in a Fusion Web Application" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

To attach policies to a connection-based Web service client:

1. Use Fusion Middleware Control to expand Application Deployments.
2. Select the target application.
3. From the **Application Deployment** menu, select **ADF**, and then **Configure ADF Connections**.
4. On the **ADF Connections Configuration** page, select a row in the **Web Service Connections** list, and then use the **Configure Web Service** list to configure the Web Service client.
5. On the Web Service Client page, select the **OWSM Policies** tab.
6. In the Directly Attached Policies section of the page, click **Attach/Detach**.
7. On the **Available Policies** section of the page, select one or more policies that you want to attach. Click **Validate** to validate the policy, or **Check Services Compatibility** to make sure that the client policies are compatible with the service policies.
8. Click **Attach** when you are sure that you want to attach the policy or policies.
9. Click **OK**.

8.4.1.3 Attaching Policies to Asynchronous Web Service Callback Clients

The following procedure describes how to attach policies to an asynchronous Web service Callback client. For more information about developing asynchronous Web services and callback clients, see "Developing Asynchronous Web Services" in *Developer's Guide for Oracle Infrastructure Web Services*.

To attach policies to an asynchronous Callback client:

1. Navigate to the endpoint for the asynchronous Web service, as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click **Callback Client** in the upper right portion of the endpoint page.
3. Click the **Policy** tab.
4. Click **Attach/Detach**.
5. On the **Available Policies** portion of the page, select one or more policies that you want to attach. Click **Validate** to validate the policy, or **Check Services Compatibility** to make sure that the client policies are compatible with the service policies.
6. Click **Attach** when you are sure that you want to attach the policy or policies.
7. Click **OK**.

8.4.1.4 Attaching Policies to Java EE Web Service Clients

This section describes how to attach a policy to a WebLogic Java EE Web service client.

Notes: For WebLogic Java EE Web service client policy attachment:

- Only Oracle WSM security policies can be attached.
- Oracle WSM policies and WebLogic Web service policies cannot be attached to the same endpoint. If a WebLogic Java EE endpoint has WebLogic policies attached, you cannot attach Oracle WSM security policies. Note that WebLogic policies can be attached using the WebLogic Server Administration Console or programmatically.
- Oracle recommends that you use Fusion Middleware Control or WLST to attach Oracle WSM policies to a Web service client post-deployment. If you attach Oracle WSM policies programmatically at development time, you will not be able to modify or delete the policies after the client application is deployed.

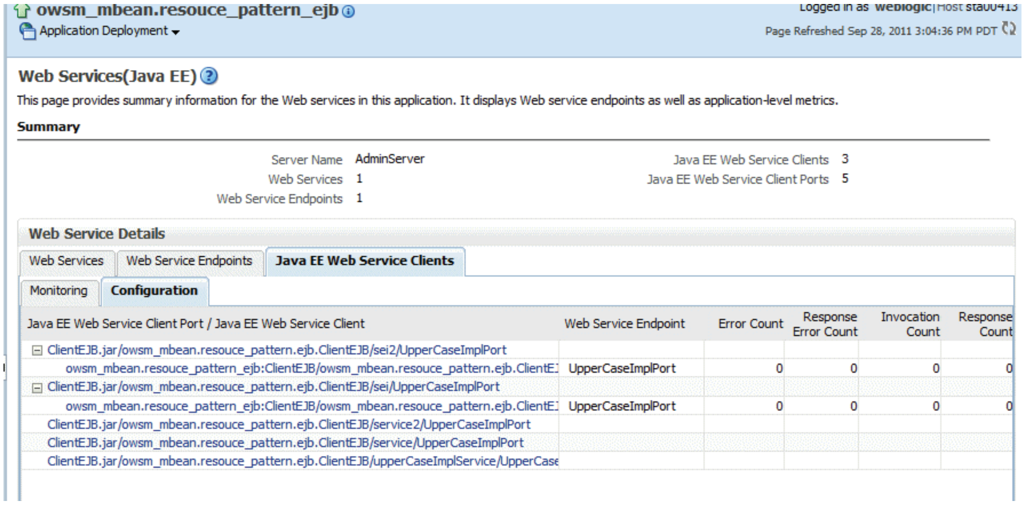
To attach a policy to a Java EE Web service client:

1. Navigate to the home page for the Java EE Web service, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. Select the **Java EE Web Service Clients** tab to view the clients in the application.
3. Select the **Configuration** tab to view the available client ports to which you can attach policies, as shown in Figure 8–8.

Notes: If a port is associated with a run-time client instance, click the + node to expand the port name to view the instance with which it is associated. You can also attach policies to ports that are defined in the client application but are not currently associated with a run-time client instance.

To ensure that the latest policies are always enforced, it is important to follow Oracle's recommended best practices when developing your WebLogic Java EE Web service client. That is, you should explicitly close client instances when processing is complete. If the client instances are not closed, any policy changes in the repository are not enforced on the client. For more information about the best practices, see "Roadmaps for Developing Web Service Clients" in *Programming Advanced Features of JAX-WS Web Services for Oracle WebLogic Server*.

Figure 8–8 Java EE Web Service Clients



owsm_mbean.resouce_pattern_ejb

Application Deployment
Logged in as weblogic | Host stau0413
Page Refreshed Sep 28, 2011 3:04:36 PM PDT

Web Services(Java EE)

This page provides summary information for the Web services in this application. It displays Web service endpoints as well as application-level metrics.

Summary

Server Name	AdminServer	Java EE Web Service Clients	3
Web Services	1	Java EE Web Service Client Ports	5
Web Service Endpoints	1		

Web Service Details

Web Services
Web Service Endpoints
Java EE Web Service Clients
Monitoring
Configuration

Java EE Web Service Client Port / Java EE Web Service Client	Web Service Endpoint	Error Count	Response Error Count	Invocation Count	Response Count
⊟ ClientEJB.jar/owsm_mbean.resouce_pattern.ejb.ClientEJB/sei2/UpperCaseImplPort					
owsm_mbean.resouce_pattern_ejb:ClientEJB/owsm_mbean.resouce_pattern.ejb.ClientE:	UpperCaseImplPort	0	0	0	0
⊟ ClientEJB.jar/owsm_mbean.resouce_pattern.ejb.ClientEJB/sei/UpperCaseImplPort					
owsm_mbean.resouce_pattern_ejb:ClientEJB/owsm_mbean.resouce_pattern.ejb.ClientE:	UpperCaseImplPort	0	0	0	0
ClientEJB.jar/owsm_mbean.resouce_pattern.ejb.ClientEJB/service2/UpperCaseImplPort					
ClientEJB.jar/owsm_mbean.resouce_pattern.ejb.ClientEJB/service/UpperCaseImplPort					
ClientEJB.jar/owsm_mbean.resouce_pattern.ejb.ClientEJB/upperCaseImplService/UpperCase					

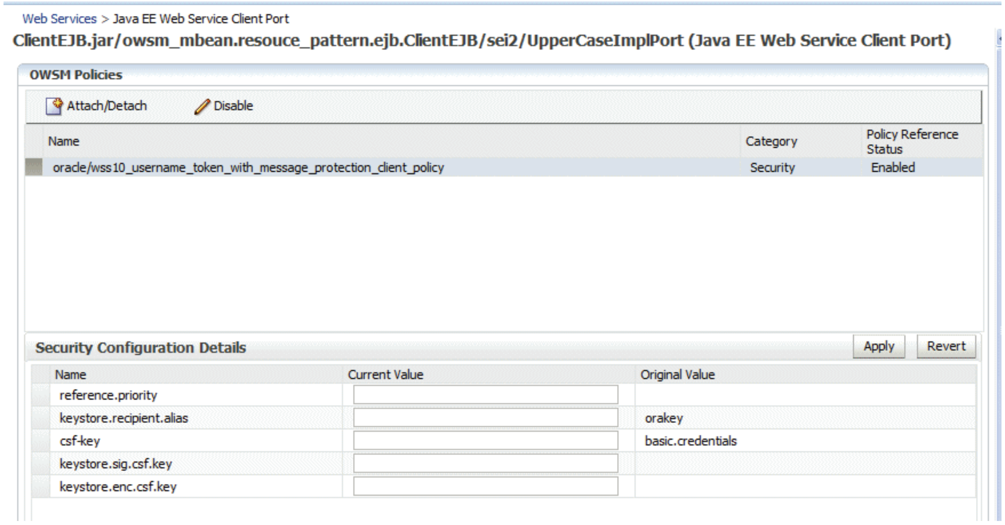
4. Click the name of the client port to navigate to the Java EE Web Service Client Port page.

5. Click **Attach/Detach**.

6. In the **Available Policies** section of the page, select one or more policies that you want to attach. Click **Validate** to verify that the combination of policies selected is valid, then click **OK**.

   The attached policy is shown on the Java EE Web Service Client Port page, as shown in

Figure 8–9 Attaching Policies to WebLogic Java EE Web Service Client Ports



Web Services > Java EE Web Service Client Port

ClientEJB.jar/owsm_mbean.resouce_pattern.ejb.ClientEJB/sei2/UpperCaseImplPort (Java EE Web Service Client Port)

OWSM Policies

Name	Category	Policy Reference Status
oracle/wss10_username_token_with_message_protection_client_policy	Security	Enabled

Security Configuration Details

Name	Current Value	Original Value
reference.priority		
keystore.recipient.alias		orakey
csf-key		basic.credentials
keystore.sig.csf.key		
keystore.enc.csf.key		

7. Optionally, specify configuration overrides for an attached policy. To do so:

   a. Select the policy for which you want to configure the overrides in the OWSM Policies section of the page.

      The properties that you can override are displayed in the Security Configuration Details section of the page.

   b. Enter the override value in the **Current Value** field for the property and click **Apply**.

   For more information about configuring overrides, see "Attaching Client Policies Permitting Overrides" on page 8-31.

8.4.2 Attaching Policies to Web Service Clients Using WLST

The following sections describe how to attach policies to Java EE and RESTful Web service clients using WLST:

- "Attaching Policies to Oracle Infrastructure and Java EE Web Service Clients Using WLST" on page 8-15
- "Attaching Policies to RESTful Web Service Clients Using WLST" on page 8-18

8.4.2.1 Attaching Policies to Oracle Infrastructure and Java EE Web Service Clients Using WLST

The following procedure describes how to attach policies to SOA references, connection-based Web service clients (such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client), Java EE Web Service Clients, and asynchronous Web service callback clients. The steps that you follow are the same for each type of client. However, the argument settings will vary depending on the type of client to which you are attaching or detaching policies.

1. View the Web service clients as described in Using WLST in "Viewing Web Service Clients" on page 6-9.

2. Use the `listWebServiceClientPorts` command to display the port name and endpoint URL for a Web service client.

```
listWebServiceClientPorts(application,moduleOrCompName,moduleType,serviceRefNam
e)
```

For example, to display the port for the service reference `client`:

```
wls:/wls-domain/serverConfig> listWebServiceClientPorts('/base_
domain/AdminServer/application1#V2.0',
'test1','wsconn','client')

HelloWorld_pt
```

3. View the list of available policies as described in "Displaying a List of the Available Policies Using WLST" on page 7-2.

   To view only available client policies, set the `subject` argument to `client`. For example:

   ```
   listAvailableWebServicePolicies("","client")
   ```

4. To attach policies, do one of the following:

   - Use the `attachWebServiceClientPolicy` command to attach a single policy to a Web service client port.

     ```
     attachWebServiceClientPolicy(application, moduleOrCompName, moduleType,
     serviceRefName, portInfoName, policyURI, [subjectType=None]
     ```

     Set the arguments as follows:

     – For a SOA reference, specify the name of the SOA composite using the `moduleOrCompName` argument, specify `soa` for the `moduleType` argument, and the name of the SOA reference using the `serviceRefName` argument.

     – For a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client, specify the name of the client application using the `application` argument, specify `wsconn` for the `moduleType` argument, and the service reference name using the `serviceRefName` argument.

     – For an asynchronous Web service callback client, specify `web` for the `moduleType` argument. Specify the name of the client application or SOA composite using the `application` and `moduleOrCompName` arguments, respectively.

     – For all client types, specify the name of the port using the `portInfoName` argument.

     – Specify the policy to be attached using the `policyURI` argument. If you specify a policy that is already attached or exists, then this command enables the policy if it is disabled.

     For example, to attach the client policy `oracle/wss_username_token_client_policy` to the `HelloWorld_pt` of the `client` service, use the following command:

     ```
     wls:/wls_domain/serverConfig> attachWebServiceClientPolicy("/wls_
     domain/AdminServer/application1#2.0",
     "test1","wsconn","client","HelloWorld_pt","oracle/wss_username_token_
     client_policy")
     ```

   - Use the `attachWebServiceClientPolicies` command to attach multiple policies to a Web service client port. Set the arguments as described for attaching a single client policy above, however you specify multiple policies to be attached using the `policyURIs` argument. If any of the policies that you specify in this command are already attached, then this command enables the policies that are already attached (if they are disabled), and attaches the others.

```
attachWebServiceClientPolicies(application, moduleOrCompName,
moduleType, serviceRefName, portInfoName, policyURIs, [subjectType=None]
```

For example, to attach the policies `oracle/wss_username_token_client_policy` and `oracle/wsrm10_policy` to the `HelloWorld_pt` of the `client` service, use the following command:

```
wls:/wls_domain/serverConfig> attachWebServiceClientPolicies("/wls_
domain/AdminServer/application1#2.0",
"test1","wsconn","client","HelloWorld_pt",
["oracle/wss_username_token_client_policy","oracle/wsrm10_policy"])

Please restart application to uptake the policy changes.
```

Note: The policyURIs are validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. If the policy validation fails, a message is displayed and the command is not executed.

If the wsm-pm application is not installed or is not available, these commands are not executed.

For additional information about validating policies, see "Validating Policy Subjects" on page 8-10.

---

5. To detach policies, do one of the following:

   - Use the `detachWebServiceClientPolicy` command to detach a single policy from a Web service client port.

     ```
     detachWebServiceClientPolicy(application, moduleOrCompName, moduleType,
     serviceRefName, portInfoName, policyURI, [subjectType=None]
     ```

     Set the arguments as described in step 4 above.

     For example, to detach the client policy `oracle/wss_username_token_client_policy` from the `HelloWorld_pt` of the `client` service, use the following command:

     ```
     wls:/wls_domain/serverConfig> detachWebServiceClientPolicy("/wls_
     domain/AdminServer/application1#2.0",
     "test1","wsconn","client","HelloWorld_pt","oracle/wss_username_token_
     client_policy")
     ```

   - Use the `detachWebServiceClientPolicies` command to detach multiple policies from a Web service client port. Set the arguments as described for detaching a single client policy above, however you specify multiple policies to be detached using the `policyURIs` argument.

     ```
     detachWebServiceClientPolicies(application, moduleOrCompName,
     moduleType, serviceRefName, portInfoName, policyURIs, [subjectType=None]]
     ```

     For example, to detach the policies `oracle/wss_username_token_client_policy` and `oracle/wsrm10_policy` from the `HelloWorld_pt` of the `client` service, use the following command:

     ```
     wls:/wls_domain/serverConfig> detachWebServiceClientPolicies("/wls_
     ```

```
domain/AdminServer/application1#2.0",
"test1","wsconn","client","HelloWorld_pt",
["oracle/wss_username_token_client_policy","oracle/wsrm10_policy"])

Please restart application to uptake the policy changes.
```

Note: When you detach a client-side security policy, you must manually remove any configuration overrides because client configuration overrides are applied at the port level. Otherwise, the override remains in effect for all future policy attachments to this port, both globally and directly.

---

6. For ADF DC and WebCenter client applications, restart the Web service client application. You do not need to restart a SOA composite.

For more information about these WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

8.4.2.2 Attaching Policies to RESTful Web Service Clients Using WLST

For information about attaching Oracle WSM policies globally to RESTful Web service clients using WLST, see "Securing RESTful Web Service Clients" in *Using the Jersey JAX-RS Reference Implementation*.

8.4.3 Enabling and Disabling Web Service Client Policies Using WLST

Use the following procedure to enable or disable policies to a Web Service client:

1. View the Web service clients as described in Using WLST in "Viewing Web Service Clients" on page 6-9.
2. Use the listWebServiceClientPorts command to display the port name and endpoint URL for a Web service client, as described in "Attaching Policies to Web Service Clients Using WLST" on page 8-15.
3. Use the enableWebServiceClientPolicy command to enable or disable a policy that is already attached to a Web Service client. Setting the command to true enables the policy. Setting it to false, disables the policy.

   ```
   enableWebServiceClientPolicy(application,moduleOrCompName,moduleType,
   serviceRefName,portInfoName,policyURI,[enable],[subjectType=None] )
   ```

   The following example enables the client policy oracle/wss_username_token_client_policy of the port JRFWssUsernamePort of the Web module WssUsernameClient. The Web service is part of the application jwsclient_1#1.1.0 for the server soa1 in the domain soainfra.

   ```
   wls:/wls-domain/serverConfig>enableWebServiceClientPolicy
   ('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
   'WssUsernameClient','JRFWssUsernamePort', "oracle/wss_username_token_client_
   policy",true)
   ```

8.5 Attaching Policies to Servlet Applications

To secure servlet applications, such as ADF business components exposed as RESTful servlets, you can attach one or more of the predefined security policies listed in Table 8–1. For more information about these policies and how to manually configure them, see Appendix B, "Predefined Policies."

> **Note:** You can also attach a SPNEGO token policy that you create using the oracle/http_spnego_token_service_template assertion template. For more information, see "Configuring Kerberos With SPNEGO Negotiation" on page 10-89.
>
> There is no client-side policy support for servlet applications in this release.

Table 8–1 Predefined Policies Supported for Servlet Applications

Predefined Policy	More Information
oracle/wss_http_token_service_policy	▪ Description of policy: "oracle/wss_http_token_service_policy" on page B-6 ▪ Description of assertion template: "oracle/wss_http_token_service_template" on page C-29 ▪ Configuring the policy: "oracle/wss_http_token_service_policy" on page 11-21
oracle/http_basic_auth_over_ssl_service_policy	▪ Description of policy: "oracle/http_basic_auth_over_ssl_service_policy" on page B-13 ▪ Description of assertion template: "oracle/wss_http_token_over_ssl_service_template" on page C-72 ▪ Configuring the policy: "oracle/http_basic_auth_over_ssl_service_policy" on page 11-36
oracle/http_jwt_token_service_policy	▪ Description of policy: "oracle/http_jwt_token_service_policy" on page B-3 ▪ Description of assertion template: "oracle/http_jwt_token_service_template" on page C-8 ▪ Configuring the policy: "oracle/http_jwt_token_service_policy" on page 11-9
oracle/http_jwt_token_over_ssl_service_policy	▪ Description of policy: "oracle/http_jwt_token_over_ssl_service_policy" on page B-14 ▪ Description of assertion template: "oracle/http_jwt_token_over_ssl_service_template" on page C-67 ▪ Configuring the policy: "oracle/http_jwt_token_over_ssl_service_policy" on page 11-39
oracle/http_oam_token_service_policy	▪ Description of policy: "oracle/http_oam_token_service_policy" on page B-3 ▪ Description of assertion template: "oracle/http_oam_token_service_template" on page C-10 ▪ Configuring the policy: "oracle/http_oam_token_service_policy" on page 11-10
oracle/http_saml20_token_bearer_service_policy	▪ Description of policy: "oracle/http_saml20_token_bearer_service_policy" on page B-4 ▪ Description of assertion template: "oracle/http_saml20_token_bearer_service_template" on page C-23 ▪ Configuring the policy: "oracle/http_saml20_bearer_token_service_policy" on page 11-19

Table 8–1 (Cont.) Predefined Policies Supported for Servlet Applications

Predefined Policy	More Information
`oracle/http_saml20_token_bearer_over_ssl_service_policy`	■ Description of policy: "oracle/http_saml20_token_bearer_over_ssl_service_policy" on page B-16 ■ Description of assertion template: "oracle/http_saml20_token_bearer_service_template" on page C-23 ■ Configuring the policy: "oracle/http_saml20_bearer_token_over_ssl_service_policy" on page 11-54
`oracle/multi_token_rest_service_policy` (OR group)	■ Description of policy: "oracle/multi_token_rest_service_policy" on page B-5 ■ Configuring the policy: "Configuring a Policy With an OR Group" on page 11-34
`oracle/multi_token_over_ssl_rest_service_policy` (OR group)	■ Description of policy: "oracle/multi_token_over_ssl_rest_service_policy" on page B-16 ■ Configuring the policy: "Configuring a Policy With an OR Group" on page 11-34
`oracle/binding_authorization_denyall_policy`	■ Description of policy: "oracle/binding_authorization_denyall_policy" on page B-37 ■ Description of assertion template: "oracle/binding_authorization_template" on page C-183 ■ Configuring the policy: "oracle/binding_authorization_denyall_policy" on page 11-102
`oracle/binding_authorization_permitall_policy`	■ Description of policy: "oracle/binding_authorization_permitall_policy" on page B-37 ■ Description of assertion template: "oracle/binding_authorization_template" on page C-183 ■ Configuring the policy: "oracle/binding_authorization_permitall_policy" on page 11-103
`oracle/binding_permission_authorization_policy`	■ Description of policy: "oracle/binding_permission_authorization_policy" on page B-38 ■ Description of assertion template: "oracle/binding_permission_authorization_template" on page C-185 ■ Configuring the policy: "oracle/binding_permission_authorization_policy" on page 11-104

For servlet applications, the Oracle WSM servlet filter is used to intercept and process the incoming request.

You can attach policies to a policy subject (servlet, in this case), either by directly attaching an individual policy to a subject, or globally attaching policies to a set of subjects by type using policy sets, as described in the following sections:

- "Attaching Policies Directly to Servlet Applications" on page 8-20
- "Attaching Policies Globally to Servlet Applications" on page 8-23

8.5.1 Attaching Policies Directly to Servlet Applications

Note: For ease of manageability, Oracle recommends that you attach the policies globally using policy sets as described in "Attaching Policies Globally to Servlet Applications" on page 8-23.

To attach policies directly to servlet applications, you must modify the `web.xml` deployment descriptor file to define the Oracle WSM servlet filter, associate it with a servlet to be secured, and define the policy attachment metadata. You can map an Oracle WSM servlet filter to a single servlet only. If you need to secure multiple servlets, you must define multiple servlet filters, maintaining a one-to-one correspondence.

For more information about the `web.xml` deployment descriptor, see "web.xml Deployment Descriptor Elements" in *Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.

To attach policies directly to servlet applications:

1. Define the Oracle WSM security filter by adding a `<filter>` element, and defining the following subelements:

   a. Specify a meaningful name for the Oracle WSM servlet filter using the `<filter-name>` element.

      For example:

      ```
      <filter>
      <filter-name>OWSM Security Filter</filter-name>
      ```

   b. Define the Oracle WSM servlet filter class using the `<filter-class>` element.

      This element must be defined as follows:

      ```
      <filter-class>
         oracle.wsm.agent.handler.servlet.SecurityFilter
      </filter-class>
      ```

   c. To pass the servlet name as a parameter to the `init()` method of the Oracle WSM servlet filter class, add an `<init-param>` element to the `<filter>` definition.

      For example:

      ```
      <init-param>
         <param-name>servlet-name</param-name>
         <param-value>TestServlet</param-value>
      </init-param>
      ```

      Note: If you omit this parameter, then the servlet application will not be protected, even if you define the `<policySet>` element in the next step.

   d. Define the security policy attachments by adding an `<init-param>` that defines a `<policySet>` element with one or more `<PolicyReference>` or `<OverrideProperty>` elements. For more information about the `<policySet>` element, see Appendix E, "Schema Reference for Policy Sets."

      Note: In this context, the `<policySet>` element does not support the `constraint` or `status` attributes. These attributes are supported for global policy attachment only.

      For example, in the following code excerpt the `<policySet>` is configured in the form of `CDATA`.

      ```
      <init-param>
         <param-name>oracle.wsm.metadata.policySet</param-name>
         <param-value><![CDATA[<sca11:policySet name="policySet"
           appliesTo="REST-Resource()"
           attachTo="Service('*')"
           xmlns:sca11="http://docs.oasis-open.org/ns/opencsa/sca/200903"
      ```

```
                xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
                xmlns:wsp15="http://www.w3.org/ns/ws-policy">
                   <wsp15:PolicyReference
                      URI="oracle/multi_token_rest_service_policy"
                      orawsp:category="security" orawsp:status="enabled">
                   </wsp15:PolicyReference>
                   <wsp15:PolicyReference
                      URI="oracle/binding_authorization_permitall_policy"
                      orawsp:category="security" orawsp:status="enabled">
                   </wsp15:PolicyReference>
                </sca11:policySet>]]>
              </param-value>
           </init-param>
```

2. Associate the Oracle WSM security filter with the servlet using the `<filter-mapping>` element.

   For example:

```
<filter>
<filter-mapping>
   <filter-name>OWSM Security Filter</filter-name>
   <servlet-name>TestServlet</servlet-name>
</filter-mapping>
```

3. Define the servlet and servlet mapping using the `<servlet>` and `<servlet-mapping>` elements.

   For example:

```
<servlet>
   <servlet-name>TestServlet</servlet-name>
   <servlet-class>webproj.TestServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>TestServlet</servlet-name>
   <url-pattern>/testservlet</url-pattern>
</servlet-mapping>
```

4. Repeat steps 1 through 3 for each servlet you wish to secure.

Example 8–1 provides an example of how to update the `web.xml` file to attach policies to a servlet application.

Example 8–1 Example of web.xml File to Attach Policies to a Servlet Applications

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         version="2.5" xmlns="http://java.sun.com/xml/ns/javaee">
   <filter>
      <filter-name>OWSM Security Filter</filter-name>
      <filter-class>oracle.wsm.agent.handler.servlet.SecurityFilter</filter-class>
      <init-param>
         <param-name>servlet-name</param-name>
         <param-value>TestServlet</param-value>
      </init-param>
      <init-param>
         <param-name>oracle.wsm.metadata.policySet</param-name>
         <param-value><![CDATA[<sca11:policySet name="policySet"
               appliesTo="REST-Resource()"
               attachTo="Service('*')"
               xmlns:sca11="http://docs.oasis-open.org/ns/opencsa/sca/200903"
               xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
               xmlns:wsp15="http://www.w3.org/ns/ws-policy">
               <wsp15:PolicyReference
                  URI="oracle/multi_token_rest_service_policy"
                  orawsp:category="security" orawsp:status="enabled">
               </wsp15:PolicyReference>
               <wsp15:PolicyReference
                  URI="oracle/binding_authorization_permitall_policy"
                  orawsp:category="security" orawsp:status="enabled">
               </wsp15:PolicyReference>
            </sca11:policySet>]]>
         </param-value>
      </init-param>
   </filter>
   <filter-mapping>
      <filter-name>OWSM Security Filter</filter-name>
      <servlet-name>TestServlet</servlet-name>
   </filter-mapping>
   <servlet>
      <servlet-name>TestServlet</servlet-name>
      <servlet-class>webproj.TestServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>TestServlet</servlet-name>
      <url-pattern>/testservlet</url-pattern>
   </servlet-mapping>
</web-app>
```

8.5.2 Attaching Policies Globally to Servlet Applications

To attach policies globally to servlet applications, you create a policy set using WLST, as described "Creating a Policy Set" on page 9-6.

When creating the policy set, ensure that the type argument is set to `REST-resource`. It is recommended that you define the resource scope as a `Domain` expression so that the global policies apply to all RESTful services in the domain.

1. Define the Oracle WSM security filter by adding a `<filter>` element, and defining the following subelements:

   a. Specify a meaningful name for the Oracle WSM servlet filter using the `<filter-name>` element.

      For example:

      ```
      <filter>
      <filter-name>OWSM Security Filter</filter-name>
      ```

   b. Define the Oracle WSM servlet filter class using the `<filter-class>` element.

      This element must be defined as follows:

      ```
      <filter-class>
         oracle.wsm.agent.handler.servlet.SecurityFilter
      </filter-class>
      ```

   c. To pass the servlet name as a parameter to the `init()` method of the Oracle WSM servlet filter class, add an `<init-param>` element to the `<filter>` definition.

For example:

```
<init-param>
   <param-name>servlet-name</param-name>
   <param-value>TestServlet</param-value>
</init-param>
```

Note: If you omit this parameter, then the servlet application will not be protected, even if you define globally attached policies.

2. Associate the Oracle WSM security filter with the servlet using the `<filter-mapping>` element.

   For example:

   ```
   <filter>
   <filter-mapping>
      <filter-name>OWSM Security Filter</filter-name>
      <servlet-name>TestServlet</servlet-name>
   </filter-mapping>
   ```

3. Define the servlet and servlet mapping using the `<servlet>` and `<servlet-mapping>` elements.

   For example:

   ```
   <servlet>
      <servlet-name>TestServlet</servlet-name>
      <servlet-class>webproj.TestServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>TestServlet</servlet-name>
      <url-pattern>/testservlet</url-pattern>
   </servlet-mapping>
   ```

4. Repeat steps 1 through 3 for each servlet you wish to secure.

Example 8–2 provides an example of attaching policies globally to servlet applications using WLST.

Example 8–2 Attaching Policies Globally to Servlet Applications Using WLST

```
C:\Oracle\Middleware\oracle_common\common\bin> wlst.cmd
...
wls:/offline> connect("weblogic","password","t3://myAdminServer.example.com:7001")
Connecting to t3://myAdminServer.example.com:7001" with userid weblogic ...
Successfully connected to Admin Server "AdminServer" that belongs to domain "my_domain".

Warning: An insecure protocol was used to connect to the
server. To ensure on-the-wire security, the SSL port or
Admin port should be used instead.

wls:/my_domain/serverConfig> beginRepositorySession()

Session started for modification.

wls:/my_domain/serverConfig> createPolicySet('myPolicySet','REST-Resource', 'Domain("*")')

Description defaulted to "Global policy attachments for REST Resource resources."
The policy set was created successfully in the session.

wls:/my_domain/serverConfig> attachPolicySetPolicy('oracle/http_basic_auth_over_ssl_service_
policy')
```

```
Policy reference "oracle/http_basic_auth_over_ssl_service_policy" added.

wls:/my_domain/serverConfig> commitRepositorySession()

The policy set myPolicySet is valid.
Creating policy set myPolicySet in repository.

Session committed successfully.

wls:/my_domain/serverConfig> displayPolicySet('myPolicySet')

Policy Set Details:
-------------------
Display Name : mypolicyset
Type of Resources:   REST Resource
Scope of Resources:  Domain("*")
Description:         Global policy attachments for REST Resource resources.
Enabled:             true
Policy Reference:    URI=oracle/http_basic_auth_over_ssl_service_policy, category=security,
enabled=true

wls:/my_domain/serverConfig>
```

8.6 Attaching Web Service Policies Permitting Overrides

> **Note:** The procedures described in this section apply to Oracle Infrastructure Web services only.

You can specify a value for server-side configuration properties in a predefined or custom Web service policy, and then either use that value each time you attach the policy to a Web service or override it on a per-attachment basis.

> **Note:** Oracle recommends that you do not edit the predefined policies so that you will always have a known set of valid policies. You can, however, create new policies using the predefined policies as a base. For additional information about creating a new policy, see "Creating a Web Service Policy from an Existing Policy" on page 7-6. Once you have created the new policy, you can edit the policy and set the configuration properties as desired.

For example, you might specify an IP address as a configuration property, and then validate the IP address from your Web service.

The scope for the server-side configuration property value is limited to the specific policy. That is, you could have two policies with the same server-side configuration property name, say *P1*, attached to the same Web service endpoint, and the two *P1* properties can have different values.

Server-side properties that you can override are of two types:

- Predefined policy configuration properties—The server-side configuration properties included with the predefined policies allow you to override certain domain-wide configuration settings from a policy, such as the CSF key used for storing the signature-key password.

- User-defined policy configuration properties—For a user-defined property, you can add a property that has meaning in your environment. You can add a user-defined server-side property to the predefined policies, or to a custom policy. For more information about creating and configuring user-defined policy configuration properties, see "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

The following sections describe how to attach Web service policies permitting overrides in more detail:

- "Configuring Server-Side Override Properties for Message Protection Policies" on page 8-26
- "Configuring Server-Side Override Properties for Authorization Policies" on page 8-28
- "Overriding Configuration Properties When Attaching a Service Policy Using Fusion Middleware Control" on page 8-29
- "Overriding Configuration Properties When Attaching a Policy Using WLST" on page 8-30

8.6.1 Configuring Server-Side Override Properties for Message Protection Policies

The predefined Oracle WSM message protection policies define the set of server-side override properties shown in Table 8–2.

If you set (or then override) these properties, the new values are used in the attached Web service instead of the keystore passwords you configure as part of setting up the keystore for message protection, as described in "Configuring Keystores for Message Protection" on page 10-9.

If you do not set these properties and leave the default blank values, the values you configure as part of setting up the keystore for message protection are used instead, as described in "Configuring Keystores for Message Protection" on page 10-9.

Table 8–2 Server-Side Configuration Properties for Message Protection Policies

Property Name	Default Value	Description
`keystore.sig.csf.key`	Blank	The alias and password used for storing the signature key password in the keystore. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. (Applicable only to WSS10 message protection policies) When used with KSS, the `keystore.sig.csf.key` and `keystore.enc.csf.key` properties in the credential store must point directly to the alias.

Table 8–2 (Cont.) Server-Side Configuration Properties for Message Protection Policies

Property Name	Default Value	Description
`keystore.enc.csf.key`	Blank	The alias and password used for storing the decryption key password in the keystore. This property allows you to specify the decryption key on a per-attachment level instead of at the domain level. (Applicable to WSS10 and WSS11 message protection synchronous policies) When used with KSS, the `keystore.sig.csf.key` and `keystore.enc.csf.key` properties in the credential store must point directly to the alias.
`saml.enveloped.signature.required`	True	Set to false (in both client and service policy) to have the bearer token be unsigned. By default (true), the bearer token is signed using the domain signature key. You can override this by using the `keystore.sig.csf.key` property in the bearer client policy.
`reference.priority`	Blank	Use to set the priority of a policy attachment in the effective policy calculation. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
`propagate.identity.context`	Blank (equivalent to False)	Set to true (in both client and service policy) to propagate the identity context from the Web service client to the Web service, and then make it available ("publish it") to other components for authentication and authorization purposes.

8.6.1.1 Setting Default Values for the Keystore Configuration Properties

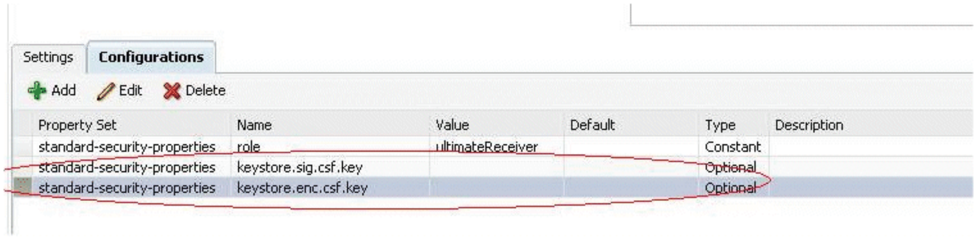
By default, the `keystore.sig.csf.key` and `keystore.enc.csf.key` properties have a blank value. You can choose to set a value such that any Web service that attaches the policy can use these values, or override the values when you attach the policy.

> **Note:** Oracle recommends that you do not edit the predefined policies so that you will always have a known set of valid policies. You can, however, create a new policy from a predefined policy and configure the properties as desired.

To set a value of a configuration property for a policy:

1. Navigate to the Web Services Policies page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the message protection policy from the Policies table and click **Edit**.
3. On the Edit Policy page, click the **Configurations** tab.
4. Select the configuration property and click **Edit** to change to the `keystore.sig.csf.key` and `keystore.enc.csf.key` properties based on the keys in your keystore. See Figure 8–10 for a partial screen.

Figure 8–10 Server-Side Configuration Properties



Property Set	Name	Value	Default	Type	Description
standard-security-properties	role	ultimateReceiver		Constant	
standard-security-properties	keystore.sig.csf.key			Optional	
standard-security-properties	keystore.enc.csf.key			Optional	

5. Validate your changes.
6. Click **Save**.

8.6.2 Configuring Server-Side Override Properties for Authorization Policies

For the predefined `oracle/binding_permission_authorization_policy` policy defines the set of server-side override properties shown in Table 8–3. You can use these properties to set a different action and resource.

If you set (or then override) these properties, the new values are used in the attached Web service instead of the action and resource you configure as described in "How Authorization Permissions Are Determined" on page 11-100.

Table 8–3 Server-Side Configuration Property for Authorization Policies

Property Name	Default Value	Description
action	*	Specify the operations for which this policy should be enforced.
resource	*	Specify the Web service name (Namespace of Web service plus ServiceName)

8.6.2.1 Setting Default Values for the Configuration Properties

By default, the *action* and *resource* properties have a value of *.   You can choose to set a different value such that any Web service that attaches the policy can use that value, or override the value when you attach the policy.

To set a value for the configuration property:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the `oracle/binding_permission_authorization_policy` policy from the Policies table and click **Edit**.

3. On the Edit Policy page, click the **Configurations** tab.
4. Select the configuration property and click **Edit** to make the change to the `action` or `resource` property based on your environment.
5. Validate your changes.
6. Click **Save**.

8.6.3 Overriding Configuration Properties When Attaching a Service Policy Using Fusion Middleware Control

After you attach a policy that includes a server-side overridable configuration property, you can then override the existing value. In WLST, you do so using the `setWebServicePolicyOverride` command as described in "Overriding Configuration Properties When Attaching a Policy Using WLST" on page 8-30.

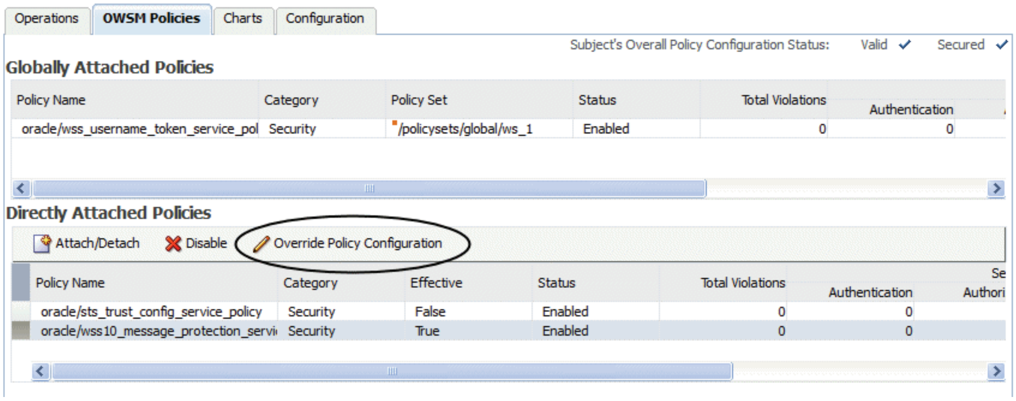
To override a configuration property using Fusion Middleware Control:

1. Select the attached policy with the overridable configuration property.

   The **Override Policy Configuration** button is displayed as shown in Figure 8–11.

> **Note:** The **Override Policy Configuration** button is displayed only when the selected policy contains configuration properties that can be overridden.

Figure 8–11 Override Policy Configuration Button



Subject's Overall Policy Configuration Status: Valid ✓ Secured ✓

Globally Attached Policies

Policy Name	Category	Policy Set	Status	Total Violations	Authentication
oracle/wss_username_token_service_pol	Security	/policysets/global/ws_1	Enabled	0	0

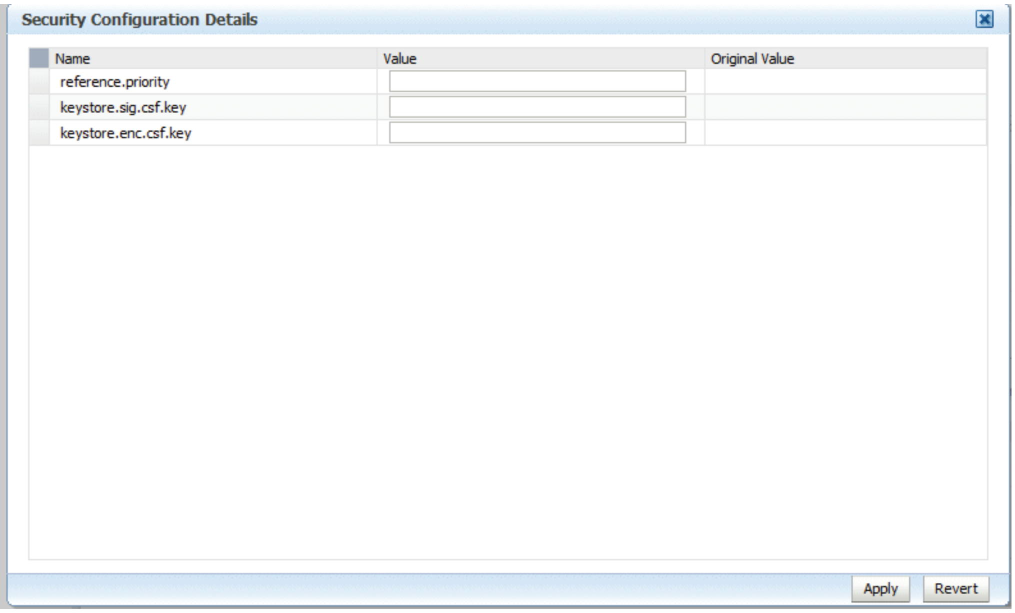
Directly Attached Policies

Policy Name	Category	Effective	Status	Total Violations	Authentication	Se Authori
oracle/sts_trust_config_service_policy	Security	False	Enabled	0	0	
oracle/wss10_message_protection_servi	Security	True	Enabled	0	0	

2. Select **Override Policy Configuration**.

   The **Security Configuration Details** window is displayed, as shown in Figure 8–12. This figure shows the overridable properties for the `oracle/wss10_message_protection_service_policy`.

Figure 8–12 Overriding a Policy Configuration Property



Name	Value	Original Value
reference.priority		
keystore.sig.csf.key		
keystore.enc.csf.key		

3. Enter the override value in the **Value** field for the property and click **Apply**.

   The property is overridden on a per-attachment basis.

For example, assume that you have not changed the value of the `keystore.sig.csf.key` property for the `oracle/wss10_message_protection_service_policy` and that it is still blank. If Web service A attaches the `oracle/wss10_message_protection_service_policy` and overrides the `keystore.sig.csf.key` property to be "sigkey," the `keystore.sig.csf.key` property has a value of "sigkey" only for the `oracle/wss10_message_protection_service_policy` attached to Web service A.

For all other policies, `keystore.sig.csf.key` uses the value you configure as part of setting up the keystore for message protection, as described in "Configuring Keystores for Message Protection" on page 10-9.

8.6.4 Overriding Configuration Properties When Attaching a Policy Using WLST

When you attach a policy that has an overridable property, you can override the existing value using the `setWebServicePolicyOverride` command. To do so, use the following procedure.

1. Attach the policy to the service as described in "Attaching a Policy to a Web Service Using WLST" on page 8-6.

2. Use the `setWebServicePolicyOverride` command to override policy properties.

   ```
   setWebServicePolicyOverride(application,moduleOrCompName,moduleType,
   serviceName,portName,policyURI,properties)
   ```

   You can override the properties listed in Table 8–2 and Table 8–3, and user-defined properties as described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

   For example, to override the `keystore.sig.csf.key` property in the `oracle/wss10_message_protection_service_policy` policy, use the following command:

```
wls:/wls-domain/serverConfig>setWebServicePolicyOverride
('/wls_domain/AdminServer/Jaxwsejb30ws','jaxwsejb',
'web','WsdlConcreteService','WsdlConcretePort',
"oracle/wss10_message_protection_service_
policy",[("keystore.sig.csf.key","sigkey")])
```

Notes: If the policy that you specify is not attached to the port, an error message is displayed and/or an exception is thrown.

If you set the `properties` argument to `None`, then all policy overrides are removed.

---

3. For ADF and WebCenter applications, restart the Web service application. You do not need to restart a SOA composite.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

For more information about this WLST command and its arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

8.7 Attaching Client Policies Permitting Overrides

The policy configuration override feature allows you to specify certain Web service client configuration information on a per-client basis, in addition to, or in lieu of setting it globally for any attachment of the policy. This targeting of configuration information limits the number of distinct policies you need to maintain.

You can define a single policy, and specify a default value for a configuration value. Rather than creating multiple policies with slightly varied configurations, you could use the same generic policy and override specific values to meet your requirements.

For example, the `oracle/wss_http_token_client_policy` policy is one example of a policy that includes the `csf-key` property, which has a default value of `basic.credentials`. The value signifies a key that maps to a username/password. It might happen that you will always use the same key value any time you attach this policy to any number of Web service clients. In this case, you can specify the key value on the `oracle/wss_http_token_client_policy` policy Configurations tab and have it apply to every instance.

However, you also have the option to override this key value on a per-client basis.

In Web service client policies, you may be able to override one or more of the properties defined in Table 8–4, depending on the policy that you attach.

If you need to clear an overridden configuration property, set it to an empty string. Before you clear it, remember that other policies could be using the same property. The properties are client-specific and there could be multiple policies that are attached to the same client that use the same property.

> **Note:** When you detach a client-side security policy, you must manually remove any configuration overrides because client configuration overrides are applied at the port level. Otherwise, the override remains in effect for all future policy attachments to this port, both globally and directly.

Table 8–4 Overridable Properties in Web Service Client Policies

Property	Notes
`attesting.mapping.attribute`	Optional, does not have to be set.
`audience.uri`	Audience restriction. Optional, does not have to be set.
`authz.code`	Optional property for passing the authorization code for the 3-legged OAuth2 use case. Not supported in this release.
`caller.principal.name`	Client's principal name as generated using the `ktpass` command and mapped to the username for which the kerberos token should be generated. Use the following format: `<username>@<REALM NAME>`. **Note:** `keytab.location` and `caller.principal.name` are required for propagating client identity for Java EE applications.
`csf-key`	Must be set on policy Configuration page or overridden.
`csf-map`	Optional, does not have to be set.
`federated.client.token`	Optional property that specifies whether a JWT token is generated for the client using the values of the oauth2.client.csf.key and keystore.sig.csf.key properties.
`include.certificate`	Optional, does not have to be set. When true, the signature certificate and the trusted certificate chain (for CA-issued certificates) are included in JWT token claim. This increases the size of the JWT token, but you do not need to then import the certificate and certificate chain into the service side keystore.
`issuer.name`	Optional, does not have to be set.
`keystore.enc.csf.key`	Optional, does not have to be set. **Note**: The `keystore.enc.csf.key` property puts the client's certificate in the replyTo header. For WSS11 policies, `keystore.enc.csf.key` is used for asynchronous clients only. For WSS10 policies, `keystore.enc.csf.key` is used for both asynchronous and synchronous clients.
`keystore.recipient.alias`	Can be set on policy Configuration page or overridden. Superseded by the Service Identity Certification Extension feature, as described in "Using Service Identity Certification Extension" on page 10-57. If the certificate is published in the WSDL, then the client override property value is ignored.
`keystore.sig.csf.key`	Optional, does not have to be set.
`keytab.location`	Location of the client's keytab file. **Note:** `keytab.location` and `caller.principal.name` are required for propagating client identity for Java EE applications.
`oauth2.client.csf.key`	Required property that specifies the key to use to obtain the client username and password. The value of oauth2.client.csf.key must match the client ID and secret expected by the client profile, as described in "Understanding OAuth Client Profiles Configuration" in *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

Table 8–4 (Cont.) Overridable Properties in Web Service Client Policies

Property	Notes
on.behalf.of	Optional, does not have to be set. Used only when sts_trust_config_client_policy is attached to a client Web service.
propagate.identity.context	Set to true (in both client and service policy) to propagate the identity context from the Web service client to the Web service, and then make it available ("publish it") to other components for authentication and authorization purposes.
redirect.uri	Optional property that specifies the redirect URIs that the OAuth server will use to redirect the user-agent to the client once access is granted or denied.
reference.priority	Optional, does not have to be set. Used to specify the priority of the policy attachment in the effective policy calculation. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
saml.assertion.filename	Optional, does not have to be set.
saml.audience.uri	Optional, does not have to be set.
saml.enveloped.signature.required	Optional, does not have to be set. Default value is true.
saml.issuer.name	Optional, does not have to be set.
scope	Optional property that specifies the scope (as-is) of the OAuth2 request. If present, the scope is included in the OAuth2 token request with this value.
service.principal.name	Must be set on policy Configuration page or overridden. Principal name for the Web service that needs to be protected, using the format <host>/<machine name>@<REALM NAME>. For example, HTTP/mymachine@EXAMPLEREALM.COM.
subject.precedence	Optional, does not have to be set. **Note**: For the wss11_saml_token_identity_switch_with_message_protection_client_policy policy, subject.precedence is required and set to false to allow for the use of a client-specified username rather than the authenticated Subject. Applications from which Oracle WSM accepts the externally-supplied identity must have the WSIdentityPermission permission. This is to avoid potentially rogue applications from providing an identity to Oracle WSM. See "Configuring Web Service Clients for Identity Switching" on page 10-78 for information about how to use subject.precedence. In particular, you need to "Directly from the Message Context" on page 10-79, and "Set the WSIdentityPermission Permission" on page 10-80. For all other SAML and JWT policies, subject.precedence is set to true and you can override it. If subject.precedence is true, the user name to create the SAML or JWT assertion is obtained only from the authenticated Subject. Similarly, if subject.precedence is false, the user name to create the SAML or JWT assertion is obtained only from the csf-key username property.
sts.auth.on.behalf.of.csf.key	Optional, does not have to be set. Used only when sts_trust_config_client_policy is attached to a client Web service.
sts.auth.user.csf.key	One or both of sts.auth.user.csf.key or sts.auth.x509.csf.key must be set, based on the STS configuration policy. Used only when sts_trust_config_client_policy is attached to a client Web service.

Table 8–4 (Cont.) Overridable Properties in Web Service Client Policies

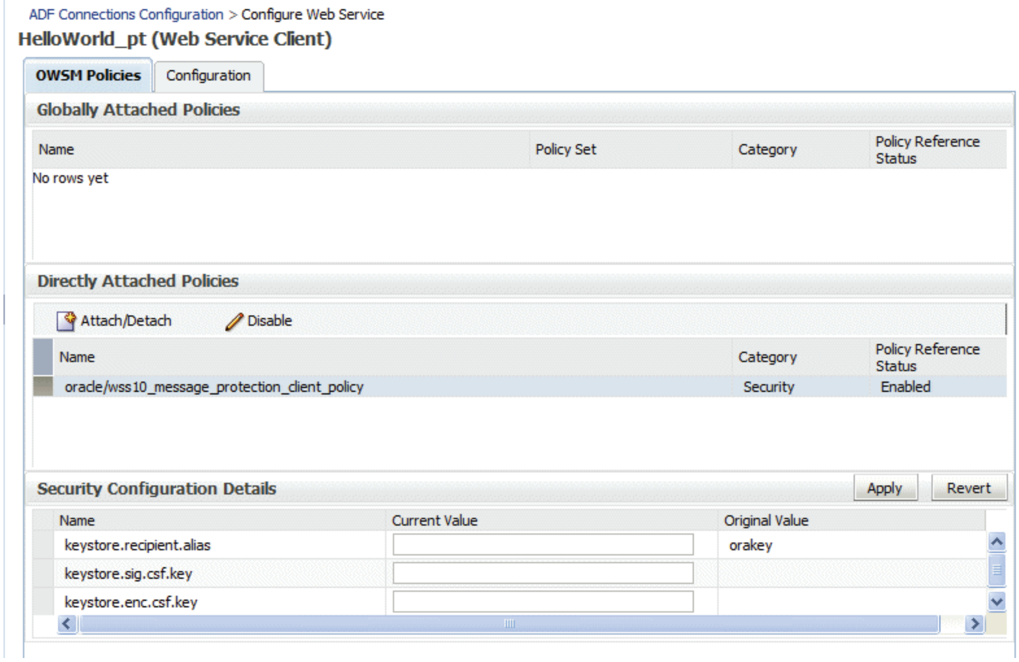
Property	Notes
`sts.auth.x509.csf.key`	One or both of `sts.auth.user.csf.key` or `sts.auth.x509.csf.key` must be set, based on the STS configuration policy. Used only when `sts_trust_config_client_policy` is attached to a client Web service.
`sts.keystore.recipient.alias`	Must be set on policy Configuration page or overridden. Used only when `sts_trust_config_client_policy` is attached to a client Web service.
`token.uri`	Required property that specifies the token endpoint of the OAuth2 server.
`user.attributes`	Optional, does not have to be set.
`user.roles.include`	Optional, does not have to be set.
`user.tenant.name`	Reserved for use with Oracle Cloud.

8.7.1 Overriding Configuration Properties When Attaching Client Policies Using Fusion Middleware Control

To override a client configuration property using Fusion Middleware Control:

1. Attach a policy to a Web service client, as described in "Attaching Policies to Web Service Clients" on page 8-11.
2. After you attach a client policy that includes a property that you can override, select the policy, then supply a value for the property in the **Security Configuration Details** section of the **OWSM Policies** page, as shown in Figure 8–13.

Figure 8–13 Overriding a Client Configuration Property



ADF Connections Configuration > Configure Web Service

HelloWorld_pt (Web Service Client)

Globally Attached Policies

Name	Policy Set	Category	Policy Reference Status
No rows yet			

Directly Attached Policies

Name	Category	Policy Reference Status
oracle/wss10_message_protection_client_policy	Security	Enabled

Security Configuration Details

Name	Current Value	Original Value
keystore.recipient.alias		orakey
keystore.sig.csf.key		
keystore.enc.csf.key		

8.7.2 Attaching Client Policies Permitting Overrides Using WLST

Note: This procedure applies to Oracle Infrastructure Web service clients only.

When you attach a client policy that has an overridable property, you can override the existing value using the `setWebServiceClientStubProperties` command.

To override a client configuration property using WLST:

1. Attach the policy to the Web service client, as described in "Attaching Policies to Web Service Clients Using WLST" on page 8-15.
2. Use the `setWebServiceClientStubProperties` command to override policy properties.

   ```
   setWebServiceClientStubProperties(application, moduleOrCompName,
    moduleType, serviceRefName, portInfoName, properties)
   ```

   For example:

   ```
   wls:soainfra/serverConfig>
   setWebServiceClientStubProperties('/soa_domain/soa_server1/adf_dc_to_bc',
    'ADF_BC', 'wsconn', 'AppModuleService', 'AppModuleServiceSoapHttpPort',
   [("csf-key","HCM_APPID")])
   ```

3. For ADF DC and WebCenter client applications, restart the Web service client application. You do not need to restart a SOA composite.

Note: You need to wait approximately 30 seconds (or the equivalent of the configured Graceful Shutdown Timeout time) between stopping and restarting the application. During this time, the server is allowing all global transactions to complete before shutting down the application. If you do not wait the configured Graceful Shutdown Timeout time, then the application will not be restarted appropriately and you will not be able to access it. To avoid waiting the graceful shutdown timeout period, you can restart the application twice.

For more information about this WLST command and its arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

8.8 Configuring User-Defined Client- or Server-Side Override Properties

Note: The procedures described in this section apply to Oracle Infrastructure Web services only.

You can use the Add New Configure Property feature to add one or more configuration properties that have meaning in your environment. Specifically, you can add one or more user-defined server- or client-side properties to the predefined policies, or to a custom policy. Then, you can either use the user-defined property as-is, or override it when you attach the policy.

In both cases, the property must already exist in the policy before you can override it when attaching the policy to a Web service or client. That is, you can override only those properties that are already present in the policy.

Therefore, you would typically add a user-supplied property with some default value to the predefined or custom policy, and then override it on a per-attachment basis.

You can add a user-defined property of type required, optional, or constant, but you cannot override a property of type constant.

The following sections describe how to configure user-defined override properties:

- "Scope of User-Defined Configuration Properties" on page 8-36
- "Adding a User-Defined Configuration Property" on page 8-36
- "Editing a User-Defined Configuration Property" on page 8-37
- "Deleting a User-Defined Configuration Property" on page 8-37
- "Overriding the Configuration Properties When Attaching a User-Defined Policy" on page 8-38

8.8.1 Scope of User-Defined Configuration Properties

As with the predefined configuration properties, the scope for user-defined configuration properties in a policy differs for clients and Web services. Consider the following:

- The scope for a client-side configuration property value is the client. There could be multiple policies that are attached to the same client that use the same property.
- The scope for a server-side configuration property value is limited to the specific policy. That is, you could have two policies with the same server-side configuration property name, say *P1*, attached to the same Web service endpoint, and the two *P1* properties can have different values.

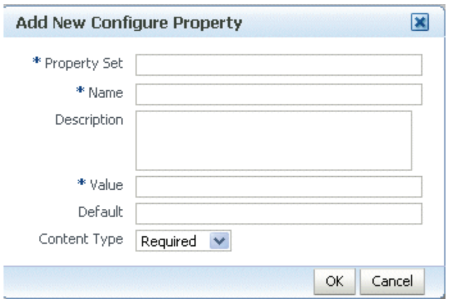
8.8.2 Adding a User-Defined Configuration Property

You edit the predefined or custom policy to add a user-defined configuration property.

To add a user-defined configuration property:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the policy for which you want to add a property from the Policies table and click **Edit**.
3. On the Edit Policy page, click the **Configurations** tab.
4. Click Add. The Add New Configure Property dialog box shown in Figure 8–14 appears.

Figure 8–14 Adding a New Configuration Property



Content Type: Required

5. Enter the following information and click **OK**.
   - **Property Set** is your name for the group (set) to which you want this property to belong. This is a required field.
   - **Name** is your name for the property. The name must be unique for this policy. This is a required field.
   - **Description** is your description for the property.
   - **Value** is the current String value for the property. This is a required field.
   - **Default** is the default String value for the property if it is not otherwise set.
   - **Content Type** can be one of Constant, Optional, or Required. You can subsequently override only properties of type Optional and Required.
6. Validate the policy.
7. Click **Save**.

8.8.3 Editing a User-Defined Configuration Property

You can edit a user-defined configuration property if you need to change it.

To edit a user-defined configuration property:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the policy for which you want to edit a property from the Policies table and click **Edit**.
3. On the Edit Policy page, click the **Configurations** tab.
4. Select the user-defined configuration property you want to edit and click **Edit**.
5. Make any needed changes.
6. Validate the policy.
7. Click **Save**.

8.8.4 Deleting a User-Defined Configuration Property

You can delete a user-defined configuration property if you no longer need it.

To delete a user-defined configuration property:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.

2. From the Web Services Policies page, select the policy for which you want to delete a property from the Policies table and click **Edit**.
3. On the Edit Policy page, click the **Configurations** tab.
4. Select the user-defined configuration property you want to delete and click **Delete**.
5. Validate the policy.
6. Click **Save**.

8.8.5 Overriding the Configuration Properties When Attaching a User-Defined Policy

Attach the user-defined policy as described in "Attaching a Policy to a Single Subject" on page 8-3, "Attaching a Policy to Multiple Subjects (Bulk Attachment)" on page 8-8, or "Attaching Policies to Web Service Clients" on page 8-11 as appropriate.

When you attach a policy that has a user-defined configuration property, you can override the existing value as follows:

- To override a user-defined configuration property for a Web service policy:
  - Using Fusion Middleware Control, see "Overriding Configuration Properties When Attaching a Service Policy Using Fusion Middleware Control" on page 8-29.
  - Using WLST, use the `setWebServicePolicyOverride` command, as described in "Overriding Configuration Properties When Attaching a Policy Using WLST" on page 8-30.
- To override a user-defined configuration property for a Web service client policy:
  - Using Fusion Middleware Control, see "Attaching Client Policies Permitting Overrides" on page 8-31.
  - Using WLST, use the `setWebServiceClientStubProperties` command, as described in "Attaching Client Policies Permitting Overrides Using WLST" on page 8-35.

9 Creating and Managing Policy Sets

Policy sets provide a means to attach policies globally to a range of endpoints of the same type. This chapter describes how to manage and create policy sets using Oracle Enterprise Manager Fusion Middleware Control and the command line interface WebLogic Scripting Tool (WLST). For information about attaching policies to policy subjects directly, see Chapter 8, "Attaching Policies to Web Services".

This chapter includes the following sections:

Notes: The procedures in this chapter apply to Oracle Infrastructure Web Services only.

To view the help for the WLST commands described in this chapter, connect to a running instance of the server and enter `help('wsmManage')`.

9.1 Understanding Global Policy Attachments Using Policy Sets

In addition to attaching policies directly to endpoints, you can create policy sets that allow you to attach policies globally to a range of endpoints of the same type, regardless of the deployment state. You can create and manage policy sets using both Fusion Middleware Control and the WebLogic Scripting Tool, WLST. Both methods are described in this chapter. Reference information about the WLST commands is provided in "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*

Attaching policies globally using policy sets allows an administrator to ensure that all subjects are secured in situations where the developer, assembler, or deployer did not explicitly specify the policies to be attached. For example, if the developer did not specify policies in annotations or include policy references in deployment descriptors, then the deployer must attach them or chance a potential security risk. By attaching policies globally to a set of subjects by type, the administrator can ensure that all subjects are secured by default independent of, and even prior to, deployment. The administrator can, for example, define a policy set that attaches a security policy to all Web service endpoints in a domain. In this case, any new services added to the domain automatically inherit the security configuration defined in the policy set. For more information, see "Determining the Secure Status of an Endpoint" on page 9-36.

Policies attached globally using policy sets also provide the following:

- The ability to specify configuration overrides on a referenced policy that apply to all endpoints to which the policy set is scoped. For information about configuring overrides, see "Overriding Configuration Properties for Globally Attached Policies" on page 9-22.
- The ability to specify a run-time constraint that determines the context in which the policy set is relevant. For example, you can specify that a service use message protection when communicating with external clients only since the message may be transmitted over insecure public networks. However, when communicating with internal clients on a trusted network, message protection may not be required. For more information, see "Specifying Run-time Constraints in Policy Sets" on page 9-26.

You can disable a globally attached policy for a specific endpoint or range of endpoints using predefined policies that do not enforce any behavior that are included with your Fusion Middleware installation. When you attach one of these policies to a specific endpoint or at a lower scope, you disable the behavior of the policy that was attached globally at the higher scope. For more information, see "Disabling a Globally Attached Policy" on page 9-30.

Policy set definitions are stored as separate XML documents in the Oracle WSM Repository under the `/policysets/global` directory.

9.1.1 Subject Types and Scope of Resources

Table 9–1, " Policy Subject Resource Types" lists the policy subjects to which you can attach OWSM policies and the valid resource scopes. For more information, see "Defining the Type and Scope of Resources" on page 9-18.

9.1.2 Typical Uses for Global Policy Attachments

Typical scenarios in which attaching policies globally can be useful include:

- All subjects of a given type need to be protected with the same set of policies, each using their default configuration. For example, all services in a domain need to be protected with authentication (using SAML or Username token) and WSS11 message protection. You can create a policy set to attach the appropriate policy to all services in the domain.
- A subset of subjects need to be protected with the same set of policies, but these policies are different from the domain-wide default. For example, all services need to be protected with authentication (using SAML or Username token), but the General Ledger application also needs stronger WSS11 message protection. You create one policy set that attaches an authentication policy to all services, and a second policy set that attaches the stronger message protection policy to the General Ledger application.
- A single subject needs to be protected by a policy in a category that is not already covered by the current set of global policy attachments and both policies need to be applied. For example, a highly-sensitive financials-based service endpoint requires permission for a client to access it in addition to the authentication and message protection required. In this case, directly attach the authorization policy to the financials-based service endpoint. The direct attachment is combined with the policies attached globally and both policies will be enforced.
- An application has been deployed with design-time policy attachments and needs to convert to using global policy attachments. The `migrateAttachments` WLST command can be used to migrate the attachments. For more information, see "Migrating Direct Policy Attachments to Global Policy Attachments" on page 9-34.

9.2 Navigating to the Policy Set Summary Page

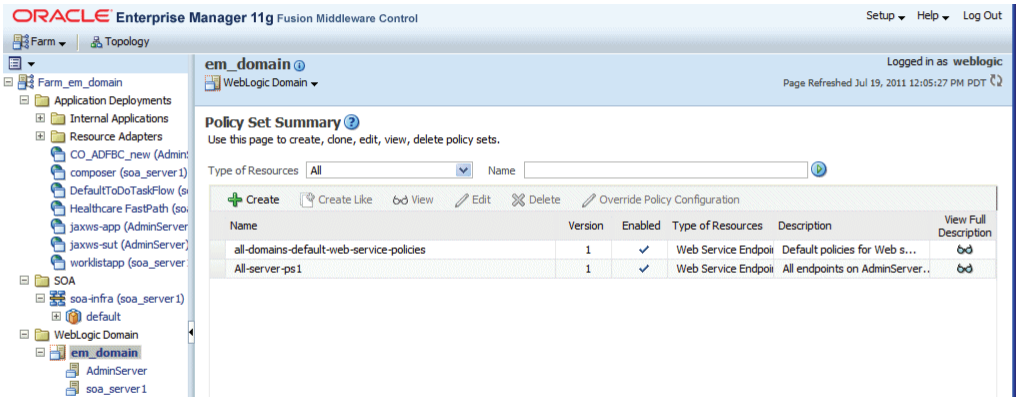
You can manage your policy sets at the domain level from the Policy Set Summary page. From this page, you can view, create, copy, edit, and delete policy sets.

To navigate to the Policy Set Summary page:

1. In the Navigator pane, expand **WebLogic Domain**.
2. Select the domain for which you want to manage policy sets.
3. From the **WebLogic Domain** menu, select **Web Services** then **Policy Sets**.

   The Policy Set Summary page is displayed, as shown in Figure 9–1.

Figure 9–1   Policy Set Summary Page



Use this page to create, clone, edit, view, delete policy sets.

Type of Resources: All

Name:

Name	Version	Enabled	Type of Resources	Description	View Full Description
all-domains-default-web-service-policies	1	✓	Web Service Endpoi	Default policies for Web s...	👓
All-server-ps1	1	✓	Web Service Endpoi	All endpoints on AdminServer..	👓

9.3 Displaying a List of Policy Sets Using WLST

To display a list of the policy sets in the repository:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Use the `listPolicySets` command to display a list of the policy sets in the repository.

   ```
   listPolicySets ([type=None])
   ```

   You can limit the display to include only those policy sets that apply to a specific type of policy subject resource types. To specify the type of subject, you must use the abbreviations specified in Table 9–1, " Policy Subject Resource Types".

   For example, to display a list of policy sets that apply to Web service endpoints:

   ```
   wls:/jrfserver_domain/serverConfig>listPolicySets('ws-service')
    Global Policy Sets in Repository:
     app-only-web-service-policies
     all-domains-default-web-service-policies
   ```

For more information about this WLST command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

9.4 Viewing the Configuration of a Policy Set

The following sections describe how to view a policy set using either Fusion Middleware Control or the command-line interface WebLogic Scripting Tool (WLST).

9.4.1 Using Fusion Middleware Control

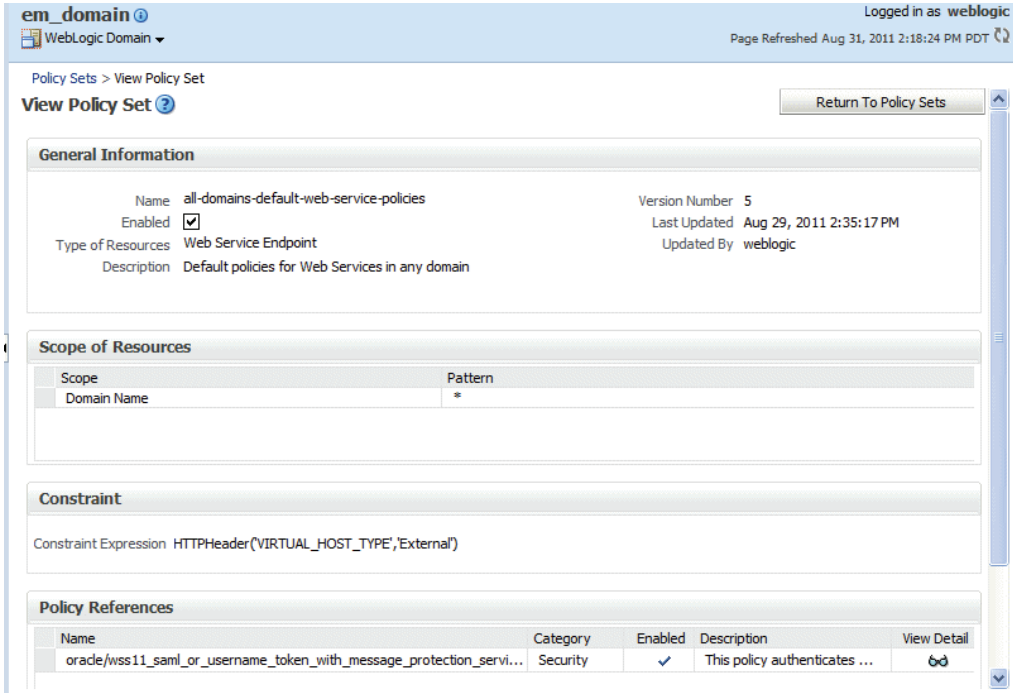
To view a policy set:

1. Navigate to the Policy Set Summary page as described in "Navigating to the Policy Set Summary Page" on page 9-3.

2. In the Policy Set Summary page, select a policy set from the table and click **View**.

3. When you are done viewing the policy set, click **Return to Policy Sets**.

Figure 9–2 Viewing a Policy Set



General Information

Name all-domains-default-web-service-policies
Enabled ☑
Type of Resources Web Service Endpoint
Description Default policies for Web Services in any domain
Version Number 5
Last Updated Aug 29, 2011 2:35:17 PM
Updated By weblogic

Scope of Resources

Scope	Pattern
Domain Name	*

Constraint

Constraint Expression HTTPHeader('VIRTUAL_HOST_TYPE','External')

Policy References

Name	Category	Enabled	Description	View Detail
oracle/wss11_saml_or_username_token_with_message_protection_servi...	Security	✓	This policy authenticates ...	👓

9.4.2 Using WLST

To view the configuration of a specific policy set in the repository:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Use the displayPolicySet command to display the configuration of a specified policy set.

   ```
   displayPolicySet ([name=None])
   ```

   When you execute this command outside of a repository session, you can display the configuration of any policy set using the name argument. If the policy set does not exist, an error message is displayed.

   If you are creating or modifying a policy set in a repository session, you do not need to specify the name argument. The current policy set is used by default. If the policy set is being modified, then the modified version is displayed. Otherwise, the latest version in the repository is displayed.

   For example:

   ```
   wls:/jrfserver_
   domain/serverConfig>displayPolicySet('int-only-web-service-policies')

    Policy Set Details:
   -------------------
   Name:                int-only-web-services-policies
   Type of Resources:   Web Service Endpoint
   Scope of Resources:  Domain("*")
   Constraint:          !HTTPHeader("VIRTUAL_HOST_TYPE","external")
   Description:         Policies for non-external client requests
   Enabled:             true
   Policy Reference:    security : oracle/wss_saml_or_username_token_service_
   ```

```
    policy, enabled=true
                          reference.priority=1
                          management : oracle/log_policy, enabled=true
```

For more information about this WLST command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

9.5 Managing Repository Modification Sessions Using WLST

When using WLST to create, modify, and delete policy sets, you must execute the commands in the context of a repository session. Each repository session applies to a single policy set only.

To create a session in which the repository will be modified, use the `beginRepositorySession` command. After you have entered the desired commands to create, modify, or delete a policy set, you write the contents of the session to the repository using the `commitRepositorySession` command.

Use the `describeRepositorySession` command to describe the contents of the current session.

To exit a repository session without writing the contents to the repository, use the `abortRepositorySession` command.

Examples of these commands are provided in the subsequent sections. For additional information, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

9.6 Creating a Policy Set

The following sections describe how to create a policy set using either Fusion Middleware Control or the command line interface WebLogic Scripting Tool, WLST.

- "Using Fusion Middleware Control" on page 9-6
- "Using WLST" on page 9-9

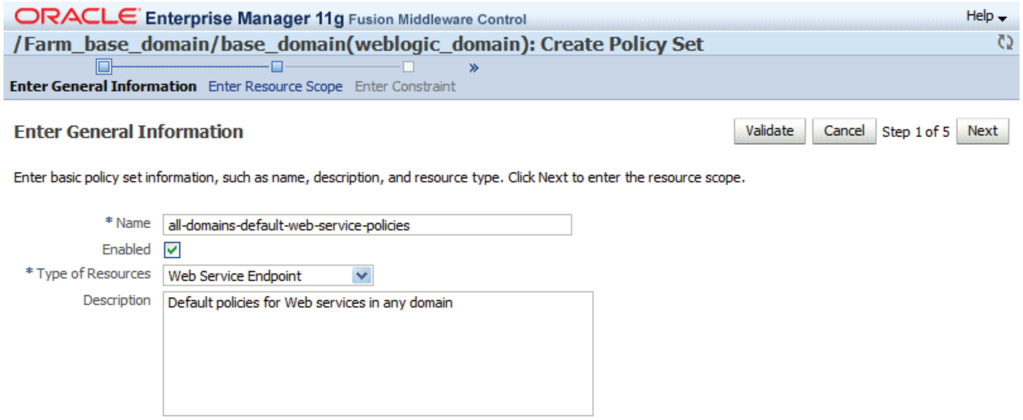
9.6.1 Using Fusion Middleware Control

To create a policy set:

1. Navigate to the Policy Set Summary page as described in "Navigating to the Policy Set Summary Page" on page 9-3.
2. From the Policy Set Summary page, click **Create**.

   The first page of the policy set creation wizard is displayed.
3. In the Enter General Information page, as shown in Figure 9–3, enter a name for the policy set.

Figure 9–3 Enter General Information Page


ORACLE Enterprise Manager 11g Fusion Middleware Control
Help
Enter General Information
Enter Resource Scope
Enter Constraint
Validate
Cancel
Step 1 of 5
Next

/Farm_base_domain/base_domain(weblogic_domain): Create Policy Set

Enter General Information

Enter basic policy set information, such as name, description, and resource type. Click Next to enter the resource scope.

* Name	all-domains-default-web-service-policies
Enabled	☑
* Type of Resources	Web Service Endpoint
Description	Default policies for Web services in any domain

4. Select the **Enabled** check box if you want to enable the policy set.

5. In the **Type of Resources** field, select the type of policy subject to which you want to attach policies. On the next page you define the scope of resources to which you want the policy set to apply. The type of policy subjects that you can select are defined in Table 9–1, " Policy Subject Resource Types".

6. Optionally, add a description of the policy set, and click **Next**.

7. In the Enter Resource Scope page, enter at least one pattern string that defines the scope for the resource type you selected in the previous step. Valid scopes are defined in Table 9–2, " Supported Expressions for the Resource Scope".

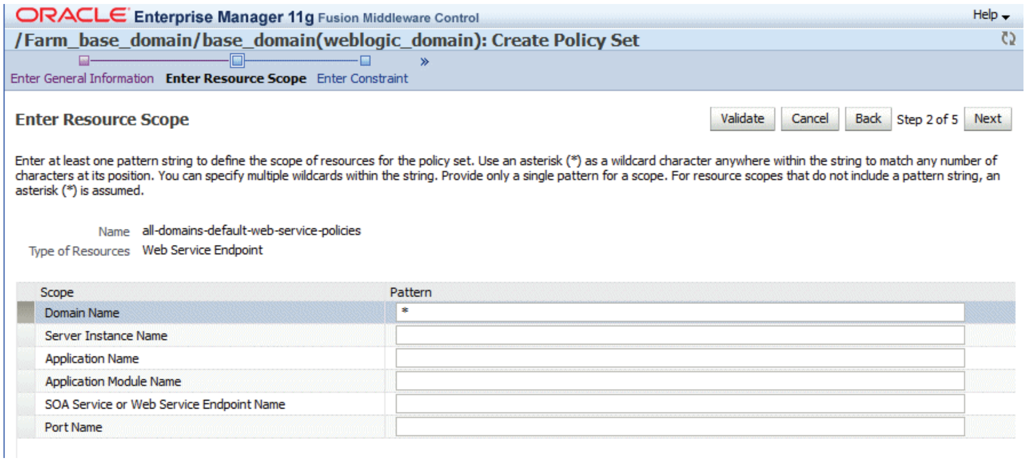
   > **Note:** To specify a resource scope, you must enter a pattern string in at least one **Pattern** field on this page.

   The list of available resource scopes is determined by the Resource Type you selected on the previous page. For example, if you selected Web Service Endpoint, the resource scopes available are Domain Name, Server Instance Name, Application Name, Application Module Name, SOA Service or Web Service Endpoint, Name and Port Name. For SOA Service resource types, the resource scopes available are Domain Name, Server Instance Name, SOA Partition Name, SOA Composite Name, SOA Service or Web Service Endpoint Name, and Port Name.

   For example, to attach the policies to all Web Service endpoints in the domain, enter a pattern string to represent the name of the domain only. You do not need to complete any of the other fields. To attach the policies at a finer scope, for example at the application or application module level, enter a pattern string to represent the name of the application or the module in the **Pattern** field. You can use an asterisk (*) as a wildcard character anywhere within the string to match any number of characters at its position; you can specify multiple wildcards within the string. Note that if you use only an asterisk wildcard for Domain, the scope level will affect *all* domains in the enterprise.

   If you provide a pattern string for multiple resource scopes, such as Domain Name and Application, the filtering conditions are ANDed together; for example, `Domain("myDomain*")` AND `Application("*myApp*")`. For more information about specifying the resource type and scope, and an example that specifies multiple resource scopes, see "Defining the Type and Scope of Resources" on page 9-18.

Figure 9–4 Enter Resource Scope Page


ORACLE Enterprise Manager 11g Fusion Middleware Control
Help
Enter General Information
Enter Resource Scope
Enter Constraint
»
Validate
Cancel
Back
Step 2 of 5
Next

/Farm_base_domain/base_domain(weblogic_domain): Create Policy Set

Enter Resource Scope

Enter at least one pattern string to define the scope of resources for the policy set. Use an asterisk (*) as a wildcard character anywhere within the string to match any number of characters at its position. You can specify multiple wildcards within the string. Provide only a single pattern for a scope. For resource scopes that do not include a pattern string, an asterisk (*) is assumed.

Name all-domains-default-web-service-policies
Type of Resources Web Service Endpoint

Scope	Pattern
Domain Name	*
Server Instance Name	
Application Name	
Application Module Name	
SOA Service or Web Service Endpoint Name	
Port Name	

8. Click **Next**.

9. In the Enter Constraint page, optionally enter a constraint to be applied to the policy set that determines the context in which the policy set is relevant. For example, you can specify that a service use message protection when communicating with external clients since the message may be transmitted over insecure public networks. However, when communicating with internal clients on a trusted network, message protection may not be required.

   To specify a constraint, in the Constraint Expression Details section of the page, select the **Enabled** check box, provide a header name and value in the **HTTP Header Name** and **HTTP Header Value** fields, optionally select the **!(NOT) Operator** to invert the constraint, and click **Update Constraint.** Then click **Next**.

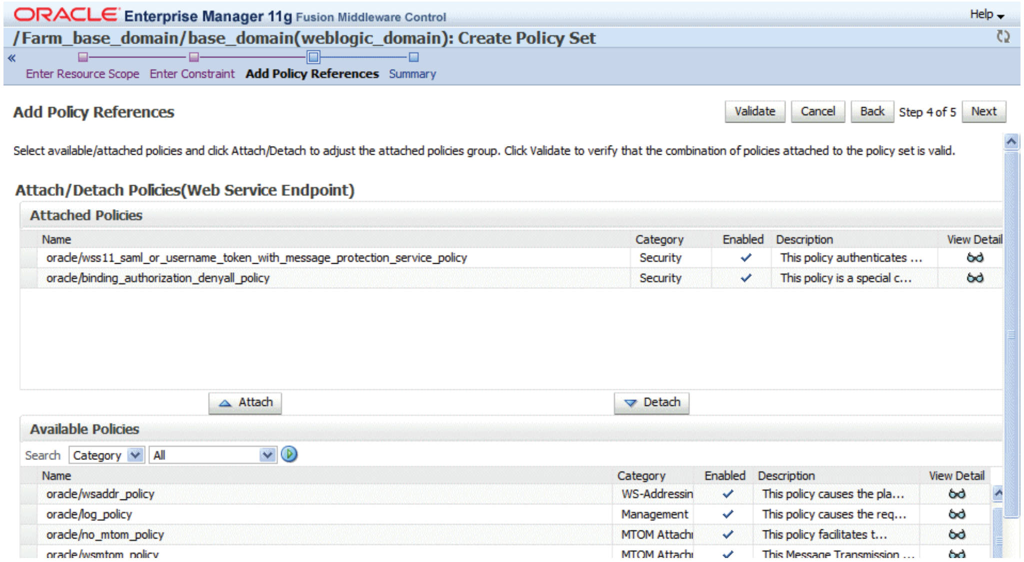
   For more information about specifying a constraint, see "Specifying Run-time Constraints in Policy Sets" on page 9-26.

10. In the Add Policy References page, select a policy from the Available Policies list, and click **Attach**.

    To view details about a policy, select the policy and click the **View Detail** icon. A pop-up window provides a full read-only description of the policy and lists the assertions that it contains. Click **OK** when you are finished reviewing the details of the policy.

11. Continue selecting and attaching policies. When you are finished, click **Validate** to verify that the combination of policies selected are valid.

Figure 9–5 Add Policy References Page



/Farm_base_domain/base_domain(weblogic_domain): Create Policy Set

Add Policy References

Select available/attached policies and click Attach/Detach to adjust the attached policies group. Click Validate to verify that the combination of policies attached to the policy set is valid.

Attach/Detach Policies(Web Service Endpoint)

Attached Policies

Name	Category	Enabled	Description	View Detail
oracle/wss11_saml_or_username_token_with_message_protection_service_policy	Security	✓	This policy authenticates ...	👓
oracle/binding_authorization_denyall_policy	Security	✓	This policy is a special c...	👓

Available Policies

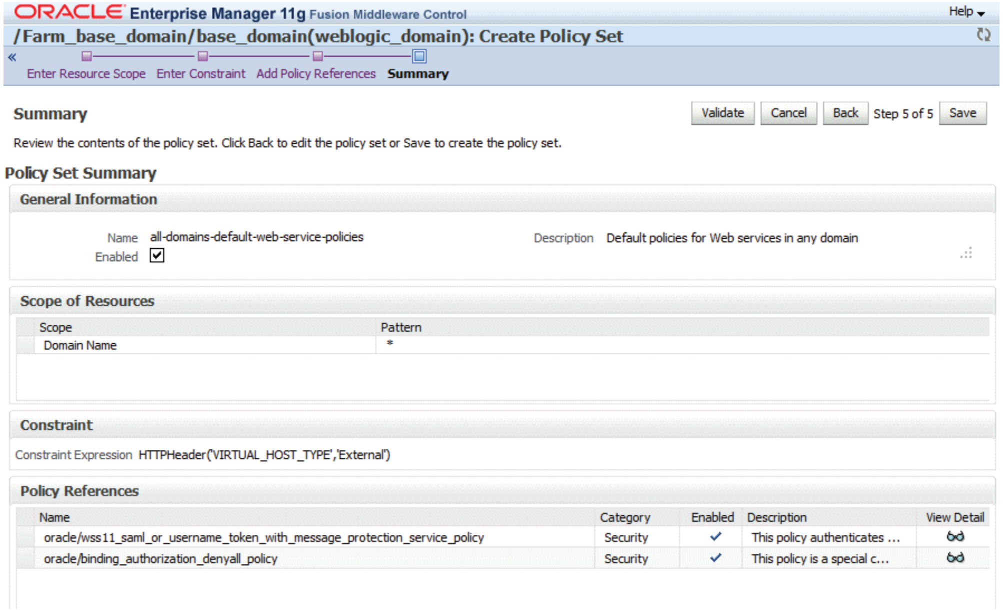
Name	Category	Enabled	Description	View Detail
oracle/wsaddr_policy	WS-Addressin	✓	This policy causes the pla...	👓
oracle/log_policy	Management	✓	This policy causes the req...	👓
oracle/no_mtom_policy	MTOM Attachi	✓	This policy facilitates t...	👓
oracle/wsmtom_policy	MTOM Attachi	✓	This Message Transmission ...	👓

12. Click **Next** to view the Policy Set Summary Page.

13. Review the policy set summary information. If you are satisfied with the policy set, click **Save**.

    Note that if the validation fails, the policy set is still saved, but in disabled mode.

Figure 9–6 Policy Set Summary Page in Create Policy Set Wizard



/Farm_base_domain/base_domain(weblogic_domain): Create Policy Set

Summary

Review the contents of the policy set. Click Back to edit the policy set or Save to create the policy set.

Policy Set Summary

General Information

Name all-domains-default-web-service-policies

Enabled ☑

Description Default policies for Web services in any domain

Scope of Resources

Scope	Pattern
Domain Name	*

Constraint

Constraint Expression HTTPHeader('VIRTUAL_HOST_TYPE','External')

Policy References

Name	Category	Enabled	Description	View Detail
oracle/wss11_saml_or_username_token_with_message_protection_service_policy	Security	✓	This policy authenticates ...	👓
oracle/binding_authorization_denyall_policy	Security	✓	This policy is a special c...	👓

9.6.2 Using WLST

Use the following procedure to create a policy set using WLST.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Begin a repository session using the `beginRepositorySession` command.

The beginRepositorySession command is used to create a session in which the repository will be modified. All creation, modification, or deletion commands must be performed in the context of a session. A session can only act on a single document.

For example:

```
wls:/jrfserver_domain/serverConfig> beginRepositorySession()

Repository session begun.
```

3. Use the createPolicySet command to create a new, empty policy set. The name, type, and attachTo arguments are required.

```
createPolicySet(name, type, attachTo, [description=None], [enable='true'])
```

Where:

- name represents the name of the new, empty policy set.
- type represents the type of policy subject to which the new policy set applies.
- attachTo represents the scope of resources to which the policy set will be attached. This argument must use a supported expression that defines a valid resource scope in a supported format. For more information, see "Defining the Type and Scope of Resources" on page 9-18.

  You do not need to enter the exact domain name for the resource scope. Wildcards are permitted, as shown in the example. For details, see "Defining the Type and Scope of Resources" on page 9-18.
- description represents an optional argument that provides a description of the policy set.
- enable specifies if the policy set is enabled or disabled. This argument is optional.

For example, to create a policy set for all services in a domain using only the required arguments:

```
wls:/jrfserver_domain/serverConfig>
createPolicySet('all-domains-default-web-service-policies', 'ws-service',
'Domain("*")')

Description defaulted to "Global policy attachments for Web Service Endpoint
resources."
The policy set was created successfully in the session.
```

Note that because no description was specified on the command line, a default description was provided.

For additional details about the arguments for this command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

4. Specify a description using the setPolicySetDescription command.

```
setPolicySetDescription(description)
```

For example, to set the description as "Default policies for web services in any domain", use the following command:

```
wls:/jrfserver_domain/serverConfig> setPolicySetDescription('Default policies
for web services in any domain')
```

```
Description updated.
```

5. To attach a policy to the current policy set, use the `attachPolicySetPolicy` command. The policy, identified by the specified URI using the `uri` argument, is attached to the endpoints specified in the policy set. You can repeat this command as needed to attach all the desired policies to the policy set.

   ```
   attachPolicySetPolicy(uri)
   ```

   For example, to attach the policy `'oracle/wss11_saml_or_username_token_with_message_protection_service_policy'` to the subjects specified in the policy set, enter the following command:

   ```
   wls:/jrfserver_domain/serverConfig>attachPolicySetPolicy('oracle/wss11_saml_or_
   username_token_with_message_protection_service_policy')

   Policy reference added.
   ```

6. Optionally, specify a configuration override or a run-time constraint. For details, refer to the following topics:

   - "Using WLST" in "Overriding Configuration Properties for Globally Attached Policies" on page 9-22.
   - "Using WLST"in "Specifying Run-time Constraints in Policy Sets" on page 9-26

7. Optionally, display the configuration of the policy set during the current repository session using the `displayPolicySet` command.

   ```
   displayPolicySet(name=None)
   ```

   Note that when you execute this command within a repository session, you do not need to specify the `name` argument. The current policy set is used by default. If the policy set is being modified, then the modified version is displayed. Otherwise, the latest version in the repository is displayed.

   For example:

   ```
   wls:/jrfserver_domain/serverConfig>displayPolicySet()

   Policy Set Details:
   -------------------
   Name:                 all-domains-default-web-service-policies
   Type of Resources:    Web Service Endpoint
   Scope of Resources:   Domain("*")
   Description:          Default policies for web services in any domain
   Enabled:              true
   Policy Reference:     security : oracle/wss11_saml_or_username_token_with_
   message_protection_service_policy, enabled=true
   ```

8. Validate the policy set using the `validatePolicySet` command.

   ```
   validatePolicySet(name=None)
   ```

   If a name is not provided, then the command validates the policy set being created or modified in the current session. Note that you can also execute this command outside of a repository session. If you do so, the `name` argument is required.

   For example:

   ```
   wls:/jrfserver_domain/serverConfig> validatePolicySet()

   The policy set all-domains-default-web-service-policies is valid.
   ```

9. Write the contents of the current repository session to the repository using the `commitRepositorySession` command.

   ```
   wls:/jrfserver_domain/serverConfig> commitRepositorySession()

   The policy set all-domains-default-web-service-policies is valid.
   Creating policy set all-domains-default-web-service-policies in repository.

   Repository session committed successfully.
   ```

   Alternately, you can choose to cancel any changes by using the `abortRepositorySession` command, which discards any changes that were made to the repository during the session.

For more information about these WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

9.7 Creating a Policy Set from an Existing Policy Set

You can use an existing policy set as the base for a new policy set. The following sections describe how to create a new policy set from an existing policy set using either Fusion Middleware Control or the command line interface WebLogic Scripting Tool, WLST.

Note that when you create a policy set from an existing policy set, all values and attachments are copied into the new one. You can modify the resource scope and the policy attachments in the new policy set, but you cannot change the type of resource to which it applies.

9.7.1 Using Fusion Middleware Control

To create a policy set using an existing policy set:

1. Navigate to the Policy Set Summary page as described in "Navigating to the Policy Set Summary Page" on page 9-3.
2. In the Policy Set Summary page, select the policy set that you want to copy and click **Create Like**.
3. In the Enter General Information page, enter a new name and description for the policy set.

   Note the following:

   - The default new policy set name is created by appending "_Copy" to the base policy set name. For example, if the base policy set is named all-domains-default-web-service-policies, the name displayed for the copy is all-domains-default-web-service-policies_Copy.
   - The Resource Type field is read-only. When you clone a policy set, you can modify the scope but not the type of resources to which the policy set will be attached.
4. Select or clear the **Enabled** check box to enable or disable the policy set.
5. Click **Next**.
6. In the Enter Resource Scope page, modify the scope as desired and click **Next**.

Note: To specify a resource scope, a pattern string must be provided in at least one **Pattern** field on this page.

---

7. In the Enter Constraint page, optionally specify a constraint or modify an existing constraint. Click **Update Constraint**, then click **Next**.

   For more information, see "Specifying Run-time Constraints in Policy Sets" on page 9-26.

8. In the Add Policy References page, modify the policy attachments as desired. When you are finished, click **Validate** to verify that the combination of polices selected is valid.

9. Click **Next** to view the Policy Set Summary Page.

10. Review the policy set summary information. If you are satisfied with the policy set, click **Save**.

9.7.2 Using WLST

To create a policy set from an existing policy set:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Begin a repository session using the `beginRepositorySession` command.

   For example:

   ```
   wls:/jrfserver_domain/serverConfig> beginRepositorySession()

   Repository session begun.
   ```

3. Use the `clonePolicySet` command to create a policy set using an existing policy set.

   ```
   clonePolicySet(name, source, [attachTo=None,] [description=None],
   [enable='true'])
   ```

   Where:

   - `name` represents the name of the new, cloned policy set.
   - `source` specifies the name of the policy set to be cloned.
   - `attachTo` represents the scope of resources to which the policy set will be attached. This argument, if provided, must use a supported expression that defines a valid resource scope in a supported format. You do not need to enter the exact name for the resource scope. Wildcards are permitted, as shown in the example. For more information, see "Defining the Type and Scope of Resources" on page 9-18.

     If this argument is not specified, then the expression used in the source policy set to identify the scope of resources is retained. You can also modify the resource scope using the `attachPolicySet` command, as described in step 5.
   - `description` represents an optional argument that provides a description of the cloned policy set.
   - `enable` specifies if the policy set is enabled or disabled. This argument is optional.

For example, to clone a policy set:

```
wls:/jrfServer_domain/serverConfig>clonePolicySet
('app-only-web-service-policies','all-domains-default-web-service-policies',
 None, 'Default policies for application jaxws-sut')

The policy set was cloned successfully in the session.
```

Note that the `attachTo` argument was not specified in this example.

For details about the arguments for this command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

4. Optionally, view the configuration of the policy set using the `displayPolicySet` command.

   For example:

```
wls:/jrfServer_domain/serverConfig> displayPolicySet()

Policy Set Details:
-------------------
Name:                 app-only-web-service-policies
Type of Resources:    Web Service Endpoint
Scope of Resources:   Domain("jrfServer_domain")
Description:          Default policies for application jaxws-sut
Enabled:              true
Policy Reference:     security : oracle/wss11_saml_or_username_token_with_
message_protection_service_policy, enabled=true
```

5. To change the resource scope of the attachments, use the `attachPolicySet` command.

```
attachPolicySet(expression)
```

   Where:

   - `expression` is a supported expression that defines the resource scope, in a supported format, that is valid for the resource type defined in the policy set. For example, for SOA resource types, you cannot define the resource scope to be an application. The supported resource scopes for SOA resource types are Domain, Server, and Composite.

     Note: Use of the Server scope is not recommended because it can cause unreliable results.

     For more information, see "Defining the Type and Scope of Resources" on page 9-18.

   For example, to attach the policies in the policy set only to the application named `jaxws-sut`, enter the following command:

```
wls:/jrfServer_domain/serverConfig> attachPolicySet('Application("jaxws-sut")')

Scope of resources updated.
```

6. Optionally, specify a configuration override or a run-time constraint. For details, refer to the following topics:

   - "Using WLST" in "Overriding Configuration Properties for Globally Attached Policies" on page 9-22.

- "Using WLST"in "Specifying Run-time Constraints in Policy Sets" on page 9-26

7. Optionally, view the configuration of the cloned policy set using the `displayPolicySet` command.

   For example:

   ```
   wls:/jrfserver_domain/serverConfig>displayPolicySet()

   Policy Set Details:
   -------------------
   Name:                 app-only-web-service-policies
   Type of Resources:    Web Service Endpoint
   Scope of Resources:   Application("jaxws-sut")
   Description:          Default policies for application jaxws-sut
   Enabled:              true
   Policy Reference:     security : oracle/wss11_saml_or_username_token_with_
   message_protection_service_policy, enabled=true
   ```

8. Write the contents of the current repository session to the repository using the `commitRepositorySession` command.

   For example:

   ```
   wls:/jrfserver_domain/serverConfig>commitRepositorySession()
   The policy set app-only-web-service-policies is valid.
   Creating policy set app-only-web-service-policies in repository.

   Repository session committed successfully.
   ```

   Alternately, you can choose to cancel any changes by using the `abortRepositorySession` command, which discards any changes that were made to the repository during the session.

For more information about these WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

9.8 Editing a Policy Set

The following sections describe how to edit an existing policy set using either Fusion Middleware Control or the command line interface WebLogic Scripting Tool, WLST.

9.8.1 Using Fusion Middleware Control

To edit an existing policy set:

1. Navigate to the Policy Set Summary page as described in "Navigating to the Policy Set Summary Page" on page 9-3.
2. In the Policy Set Summary page, select the policy set that you want to edit and click **Edit**.
3. In the Enter General Information page, select or clear the **Enabled** check box to enable or disable the policy set. You can also edit the policy set description.

   Note that the Name and Type of Resources fields are read-only.
4. Click **Next**.
5. In the Enter Resource Scope page, modify the scope as desired and click **Next**.
6. In the Enter Constraint page, optionally specify a constraint or modify an existing constraint. Click **Update Constraint**, then click **Next**.

For more information about specifying a constraint, see "Specifying Run-time Constraints in Policy Sets" on page 9-26.

7. In the Add Policy References page, modify the policy attachments as desired. When you are finished, click **Validate** to verify that the combination of polices selected is valid.
8. Click **Next** to view the Policy Set Summary Page.
9. Review the policy set summary information. If you are satisfied with the policy set, click **Save**.

9.8.2 Using WLST

To edit a policy set:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Begin a repository session using the `beginRepositorySession` command.

   For example:

   ```
   wls:/jrfserver_domain/serverConfig> beginRepositorySession()

   Repository session begun.
   ```

3. Use the `modifyPolicySet` command to select an existing policy set to edit.

   ```
   modifyPolicySet(name)
   ```

   The latest version of the named policy set will be loaded into the current session. For example, to edit a policy set to add policies, use the following command:

   ```
   wls:/jrfServer_domain/serverConfig>
   modifyPolicySet('all-domains-default-web-service-policies')

   The policy set is ready for modification in the session.
   ```

4. Edit the policy set as desired. For example:

   - To add policies to the policy set, use the `attachPolicySetPolicy` command, identifying the policy by a specified URI using the `uri` argument.

     ```
     attachPolicySetPolicy(uri)
     ```

     To add the `oracle/wss_saml_or_username_token_service_policy` and the `oracle/log_policy` policies to the policy set, enter the following commands:

     ```
     wls:/jrfServer_domain/serverConfig> attachPolicySetPolicy('oracle/wss_saml_
     or_username_token_service_policy')

     Policy reference added.

     wls:/jrfServer_domain/serverConfig>attachPolicySetPolicy('oracle/log_
     policy')

     Policy reference added.
     ```

   - To remove policies from the policy set, use the `detachPolicySetPolicy` command, identifying the policy by a specified URI using the `uri` argument.

     ```
     detachPolicySetPolicy(uri)
     ```

To remove the oracle/wss11_saml_or_username_token_with_message_protection_service_policy from the policy set, enter the following:

```
wls:/jrfServer_domain/serverConfig> detachPolicySetPolicy('oracle/wss11_
saml_or_username_token_with_message_protection_service_policy')

Policy reference removed.
```

- To enable or disable a policy attachment in the policy set, use the enablePolicySetPolicy command, identifying the policy by a specified URI using the uri argument.

  ```
  enablePolicySetPolicy(uri,[enable=true])
  ```

  The default is true.

  To disable the oracle/log_policy, enter the following:

  ```
  wls:/jrfServer_domain/serverConfig> enablePolicySetPolicy('oracle/log_
  policy',false)

  Policy reference disabled.
  ```

5. Optionally, specify a configuration override or a run-time constraint. For details, refer to the following topics:

   - "Using WLST" in "Overriding Configuration Properties for Globally Attached Policies" on page 9-22.
   - "Using WLST"in "Specifying Run-time Constraints in Policy Sets" on page 9-26

6. Validate the policy set using the ValidatePolicySet command.

   For example:

   ```
   wls:/jrfServer_domain/serverConfig> validatePolicySet()

   The policy set app-only-web-service-policies is valid.
   ```

7. Optionally, display the modified policy set using the displayPolicySet command.

   ```
   wls:/jrfServer_domain/serverConfig>displayPolicySet()

   Policy Set Details:
   -------------------
   Name:                 all-domains-default-web-service-policies
   Type of Resources:    Web Service Endpoint
   Scope of Resources:   Domain("*")
   Description:          Default policies for web services in any domain
   Enabled:              true
   Policy Reference:     security : oracle/wss_saml_or_username_token_service_
   policy, enabled=true
                         management : oracle/log_policy, enabled=false
   ```

8. To write the contents of the current repository session to the repository, use the commitRepositorySession command.

   ```
   wls:/jrfServer_domain/serverConfig> commitRepositorySession()

   The policy set all-domains-default-web-service-policies is valid.
   Updating policy set all-domains-default-web-service-policies in repository.

   Repository session committed successfully.
   ```

Alternately, you can choose to cancel any changes by using the `abortRepositorySession` command, which discards any changes that were made to the repository during the session.

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

9.9 Defining the Type and Scope of Resources

The resource type, or subject type, identifies the type of endpoint to which the policy set applies. It is mapped to the `appliesTo` attribute of the policy set.

The resource scope expression identifies the set of policy subjects to which the policy set is attached (which can be zero, one, or many). It is mapped to the `attachTo` attribute of the policy set. and is used for conflict resolution when multiple policy sets exist. For details about conflict resolution in policy sets, see "How the Effective Set of Policies is Calculated" on page 9-38.

To attach policies globally across a set of resources, you must specify the type of policy subjects to which the policy set applies and the scope of resources within the topology of the enterprise.

9.9.1 Resource Type

In Fusion Middleware Control, you select the resource type from a menu when you are creating a policy set. When you create a policy set using WLST, you must use specific abbreviations for these resource types. Table 9–1 lists the type of resources that you select in Fusion Middleware Control, the abbreviations that are required in WLST, and the resource scopes that are valid for each resource type.

> **Note:** Use of the Server Instance scope is not recommended because it can cause unreliable results, especially in applications that are targeted to multiple servers, including clusters.

Table 9–1 Policy Subject Resource Types

Fusion Middleware Control	WLST	Valid Resource Scope
ADF RESTful Web Service Connection	Reserved for future use.	Reserved for future use.
Asynchronous Callback Client	ws-callback	▪ Domain ▪ Application ▪ Application Module ▪ Web Service Endpoint ▪ Port
RESTful Client	rest-client	▪ Domain ▪ Application ▪ Module

Table 9–1 (Cont.) Policy Subject Resource Types

Fusion Middleware Control	WLST	Valid Resource Scope
RESTful Resource	rest-resource	■ Domain ■ Application ■ Module ■ Service **Note**: Server Instance is not a valid resource scope for RESTful resources.
SOA Component	sca-component	■ Domain ■ SOA Partition ■ SOA Composite ■ SOA Component
SOA Reference	sca-reference	■ Domain ■ SOA Partition ■ SOA Composite ■ SOA Reference ■ Port
SOA Service	sca-service	■ Domain ■ SOA Partition ■ SOA Composite ■ SOA Service ■ Port
Web Service Client	ws-client	■ Domain ■ Application ■ Application Module ■ Web Service Client ■ Port
Web Service Connection	ws-connection	■ Domain ■ Application ■ Application Module ■ SOA Reference or Web Service Client ■ Port
Web Service Endpoint	ws-service	■ Domain ■ Application ■ Application Module ■ Web Service Endpoint ■ Port

9.9.2 Resource Scope

In Fusion Middleware Control, you specify the scope by entering a pattern string that represents the name associated with the resource scope. For example, to attach a policy set to all Web service endpoints in a domain, you enter a pattern that represents the name of the domain in the **Domain Name** field.

When specifying the resource scope in WLST, you need to use a supported expression for each scope. The supported expressions are described in Table 9–2. These expressions are required for the following arguments:

- `attachTo` argument of the `createPolicySet` and `clonePolicySet` commands
- `expression` argument of the `attachPolicySet` command

For both Fusion Middleware Control and WLST, you can enter the complete name, or a partial value using wildcards. An asterisk (*) is permitted as a wildcard character anywhere within the string to match any number of characters at its position. You can specify multiple wildcards at any position within the string. For example, for the domain name `jrf_domain`, you can enter jrf*, or *rf*domain, or any number of combinations. You need to provide only a single pattern for a scope. If you do not specify a pattern string for a resource scope, asterisk (*) is assumed. You can use single or double quotes. If multiple values are provided, then all of the expressions must match for the policy set to be considered attached to the policy subject.

The following is a list of the supported expressions for the resource scope.

Table 9–2 Supported Expressions for the Resource Scope

Supported Expression	Description
Domain("expression")	This value will be matched against a policy subject based on the management domain in which it is deployed.
Server("expression")	This value will be matched against a policy subject based on the server instance in which it is deployed. **Note:** Use of this scope is not recommended because it can cause unreliable results, especially in applications that are targeted to multiple servers, including clusters.
Application("expression")	This value will be matched against a policy subject based on the name of the application in which it is located.
Partition('expression")	This value will be matched against a policy subject based on the name of the partition in which it is located.
Module("expression")	This value will be matched against a policy subject based on the name of the application module in which it is located.
Composite("expression")	This value will be matched against a policy subject based on the name of the SOA composite in which it is located. **Note:** For a composite, the expression should use the composite name only, for example: `Composite("*Basic_SOA_Client*")` Do not include the SOA partition or composite revision number in the expression.
Reference("expression")	This value will be matched against a policy subject based on the name of the reference in which it is located.

Table 9–2 (Cont.) Supported Expressions for the Resource Scope

Supported Expression	Description
Service("expression")	This value will be matched against a policy subject based on the name of the service in which it is located. **Note**: For Web Service Endpoint and Web Service Client resource types (ws-service and ws-client), the expression must include the namespace and the service name, for example: `Service("{http://mynamespace/}myService")` For applications assembled prior to PS5, the namespace is not displayed in the `listWebServices` output or in Fusion Middleware Control where the service name is displayed. In this case, you can determine the namespace as described in "Determining the Namespace for a Web Service" on page 9-21.
Component("expression")	This value will be matched against a policy subject based on the name of the component in which it is located.
Port("expression")	This value will be matched against a policy subject based on the name of the port in which it is located.

9.9.3 Determining the Namespace for a Web Service

For applications assembled prior to PS5, the namespace is not displayed with the service name in the output for WLST commands, or in Fusion Middleware Control where the service name is displayed. To specify a service as a resource scope for Web Service Endpoints and Web Service Client resource types (ws-service and ws-client), you need to include the namespace with the service name. You can determine the namespace for a service from the Web service WSDL document. To do so:

1. Display the WSDL document for the Web service endpoint as described in "Displaying the Web Service WSDL Document" on page 6-12.
2. In the WSDL document, locate the `wsdl:definitions` element, which includes the target namespace for the service.

   For example, in the TestService WSDL:

   ```
   http://host:7001/jaxws-service/TestService?WSDL
   ```

   The following `wsdl:definitions` element is included:

   ```
   <wsdl:definitions name="TestService"
   targetNamespace="http://service.jaxws.wsm.oracle/">
   ```

To specify a complete service name, combine the namespace with the service name. Using the example above, the complete service name is as follows:

```
{http://service.jaxws.wsm.oracle/}TestService
```

9.9.4 Examples

The following examples demonstrate how to create policy sets using different resource types and scopes.

Example 9–1 creates a policy set for an asynchronous callback client (ws-callback) resource type. In this example, the policy set is attached at a specific application scope, and applies to all services that satisfy the filter condition (`Domain` AND `Application`).

Example 9–1 Asynchronous Callback Client Resource Type Policy Set

```
beginRepositorySession()
createPolicySet('Async callback client', 'ws-callback',
 'Domain('FinancialDomain') and Application('Expense*')',
'Global policy for asynchronous callback client', true)
attachPolicySetPolicy('oracle/wss10_saml_token_client_policy')
validatePolicySet()
commitRepositorySession()
displayPolicySet('Async callback client')
```

Example 9–2 creates a policy set named web_connection_cost_service for a Web service connection (ws-connection) resource type. In this example, the policy set is attached at a specific application module scope, and applies to all services that satisfy the filter condition (`Domain` AND `Application` AND `Module`).

Example 9–2 Web Service Connection Resource Type Policy Set

```
beginRepositorySession()
createPolicySet('web_connection_cost_service', 'ws-connection',
 'Domain("SCMDomain") and and Application("ScmCst*") and Module("*Costs")',
enable=true)
attachPolicySetPolicy('oracle/wss10_saml_token_client_policy')
validatePolicySet()
commitRepositorySession()
displayPolicySet('web_connection_cost_service')
```

9.10 Validating a Policy Set

In addition to validating that the policy set adheres to the rules described in "Validating Policy Subjects" on page 8-10, policy set validation also performs the following checks:

- Validates that the defined resource type and scope is valid for the policy set
- Validates that the value entered for the resource scope contains a supported expression in a supported format
- Validates that any referenced policies are available and compatible with each other. For example, the policies are compatible if their categories are not in conflict with each other.

> **Note:** To ensure there are no conflicts between policy attachments, you can use Fusion Middleware Control and WLST commands to determine if Web service endpoints contain a valid and secure configuration. For more information, see "Determining the Secure Status of an Endpoint" on page 9-36.
>
> For troubleshooting information, see "Diagnosing Policy Attachment Issues Using WLST" on page 16-15.

9.11 Overriding Configuration Properties for Globally Attached Policies

If a policy referenced in a policy set contains overridable properties, you can override the existing value of the property for that policy set using either Fusion Middleware Control or WLST.

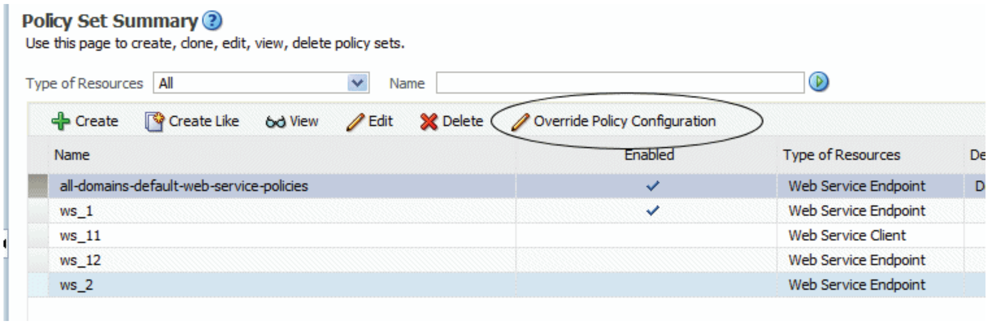
9.11.1 Using Fusion Middleware Control

To override a configuration property in a policy referenced in a policy set:

1. Go to the Policy Set Summary page as described in "Navigating to the Policy Set Summary Page" on page 9-3.
2. From the Policy Set Summary page, select the policy set containing the policy for which you want to configure overrides.

   If the policy set references policies with overridable properties, the **Override Policy Configuration** button is displayed, as shown in Figure 9–7.

Figure 9–7 Policy Set Override Policy Configuration Button



Policy Set Summary

Use this page to create, clone, edit, view, delete policy sets.

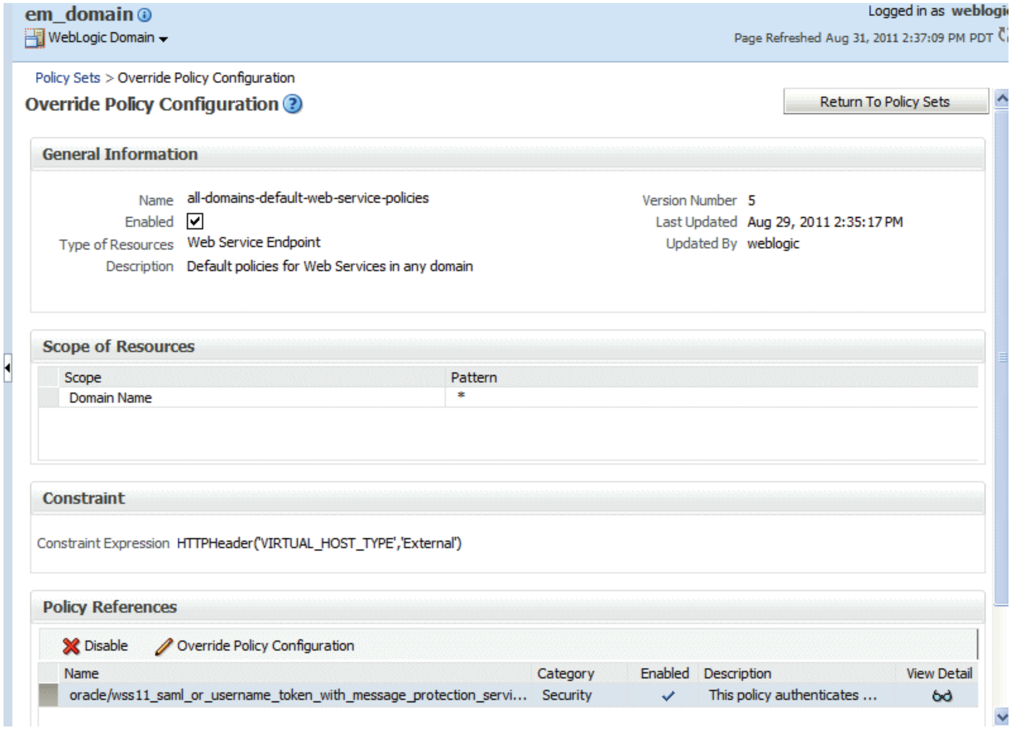
Type of Resources: All   Name:

Name	Enabled	Type of Resources	De
all-domains-default-web-service-policies	✓	Web Service Endpoint	D
ws_1	✓	Web Service Endpoint	
ws_11		Web Service Client	
ws_12		Web Service Endpoint	
ws_2		Web Service Endpoint	

3. Select **Override Policy Configuration**.

   The Override Policy Configuration page is displayed, as shown in Figure 9–8.

Figure 9–8 Policy Set Override Policy Configuration Page



em_domain

WebLogic Domain

Logged in as weblogi

Page Refreshed Aug 31, 2011 2:37:09 PM PDT

Policy Sets > Override Policy Configuration

Override Policy Configuration

General Information

Name: all-domains-default-web-service-policies

Enabled: ☑

Type of Resources: Web Service Endpoint

Description: Default policies for Web Services in any domain

Version Number: 5

Last Updated: Aug 29, 2011 2:35:17 PM

Updated By: weblogic

Scope of Resources

Scope	Pattern
Domain Name	*

Constraint

Constraint Expression: HTTPHeader('VIRTUAL_HOST_TYPE','External')

Policy References

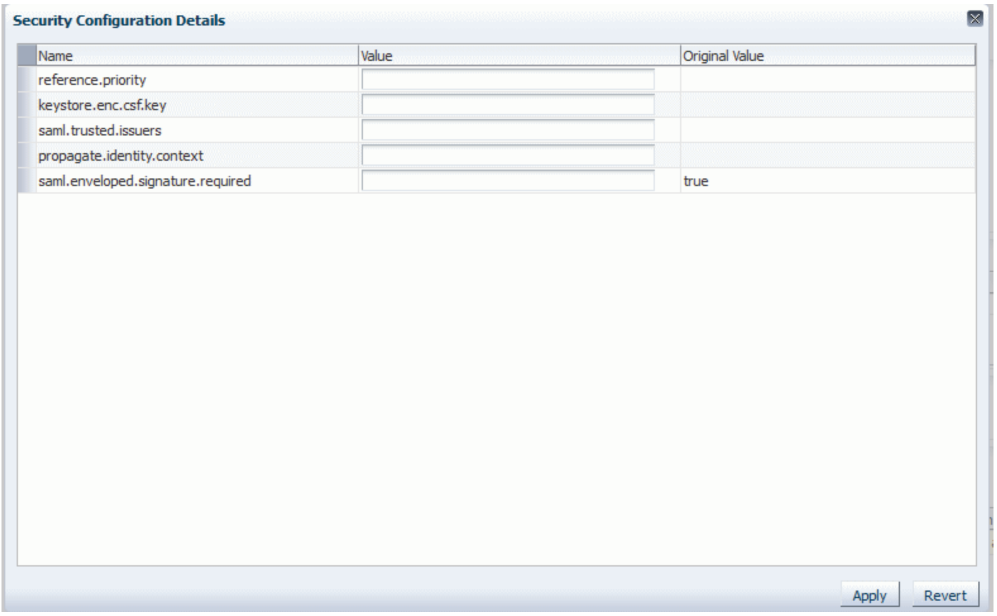
Name	Category	Enabled	Description	View Detail
oracle/wss11_saml_or_username_token_with_message_protection_servi...	Security	✓	This policy authenticates ...	👓

4. In the Policy References table, select the policy for which you want to override the configuration property. If the policy contains overridable properties, the **Override Policy Configuration** button is displayed.

5. Select **Override Policy Configuration**. The Security Configuration Details page is displayed, containing a list of the configuration properties that can be overridden in the selected policy.

   Figure 9–9 shows the overridable properties for the `oracle/wss11_saml_or_username_token_with_message_protection_service_policy`.

Figure 9–9 Policy Set Security Configuration Details Page



Name	Value	Original Value
reference.priority		
keystore.enc.csf.key		
saml.trusted.issuers		
propagate.identity.context		
saml.enveloped.signature.required		true

6. Enter the override value in the **Value** field for the property and click **Apply**.

   The property will be overridden for all endpoints to which the policy set applies.

For more information about configuration overrides and the properties that can be overridden, see the following sections:

- "Attaching Web Service Policies Permitting Overrides" on page 8-25
- "Attaching Client Policies Permitting Overrides" on page 8-31
- "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35

9.11.2 Using WLST

You can specify a configuration override in a policy referenced in a policy set using the `setPolicySetPolicyOverride` command. This command can be used only during the creation or modification of a policy set within the context of a repository session. The following procedure describes how to specify a configuration override while editing an existing policy set, but you can also use this command in a repository session while creating a new policy set or creating a policy set from an existing policy set.

1. Begin a repository session using the `beginRepositorySession` command.

   For example:

```
wls:/jrfserver_domain/serverConfig> beginRepositorySession()

Repository session begun.
```

2. Use the `modifyPolicySet` command to select an existing policy set to edit.

   ```
   modifyPolicySet(name)
   ```

   The latest version of the named policy set will be loaded into the current session. For example, enter the following command:

   ```
   wls:/jrfServer_domain/serverConfig>
   modifyPolicySet('all-domains-default-web-service-policies
   ')

   The policy set is ready for modification in the session.
   ```

3. Optionally, view the configuration of the policy set using the `displayPolicySet` command.

   For example:

   ```
   wls:/jrfserver_domain/serverConfig>displayPolicySet()

   Policy Set Details:
   -------------------
   Name:                 all-domains-default-web-service-policies
   Type of Resources:    Web Service Endpoint
   Scope of Resources:   Domain("*")
   Description:          Default policies for web services in any domain
   Enabled:              true
   Policy Reference:     security : oracle/wss_saml_or_username_token_service_
   policy, enabled=true
                         management : oracle/log_policy, enabled=false
   ```

4. Specify the configuration override using the `setPolicySetPolicyOverride` command.

   For example, to specify a configuration override for the r`eference.priority` property, enter the following command:

   ```
   wls:/jrfserver_domain/serverConfig>setPolicySetPolicyOverride('oracle/wss_saml_
   or_username_token_service_policy',
   'reference.priority','1')

   The configuration override property "reference.priority" having value "1" has
   been added to the reference to policy with URI "oracle/wss_saml_or_username_
   token_service_policy".
   ```

5. Optionally, view the configuration of the policy set.

   For example:

   ```
   wls:/jrfserver_domain/serverConfig>displayPolicySet()

   Policy Set Details:
   -------------------
   Name:                 all-domains-default-web-service-policies
   Type of Resources:    Web Service Endpoint
   Scope of Resources:   Domain("*")
   Description:          Default policies for web services in any domain
   Enabled:              true
   Policy Reference:     security : oracle/wss_saml_or_username_token_service_
   ```

```
policy, enabled=true
                     reference.priority=1
                     management : oracle/log_policy, enabled=false
```

Note that the `reference.priority` configuration override is now shown in the output (in bold in the above example.)

6. Validate the policy set using the `ValidatePolicySet` command.

   For example:

   ```
   wls:/jrfServer_domain/serverConfig> validatePolicySet()

   The policy set all-domains-default-web-service-policies is valid.
   ```

7. To write the contents of the current repository session to the repository, use the `commitRepositorySession` command.

   ```
   wls:/jrfServer_domain/serverConfig> commitRepositorySession()

   The policy set all-domains-default-web-service-policies is valid.
   Updating policy set all-domains-default-web-service-policies in repository.

   Repository session committed successfully.
   ```

9.12 Specifying Run-time Constraints in Policy Sets

Applications can be deployed into environments that expose the same services to both external and internal clients. In these environments, it is often desirable to enforce different security behaviors based on where the client is located.

For example, in an environment consisting of a single Fusion Middleware server (WebLogic Server) hosting Web services, Oracle HTTP Server is configured at the front end to listen for HTTP requests from two separate networks. One of these networks is used to transport all private internal requests, while the other network is used to transport all external requests. Access via the external network is through a firewall. Since physical access to the internal network is highly restricted, requests from this network are already protected. Therefore, it is only necessary to enforce authentication and authorization. By not enforcing message protection, the load on the server is reduced and performance is increased. However, all requests from the external network are considered to be insecure since it can be potentially accessed by anyone. In this case, in addition to authentication and authorization, message protection (confidentiality and integrity) must be enforced. Performance for these requests will be lower, but is considered acceptable since the alternatives (such as data leaks, replay attacks, and so on) are far worse.

To ensure that a policy set is applied appropriately for an external network, the administrator needs to specify a constraint expression against which the policy set is evaluated. The value of the expression indicates the runtime context for which the policy set is relevant.

The constraint expression must specify a valid header name and value. The following expressions are certified in this release:

- `HTTPHeader("VIRTUAL_HOST_TYPE","External")`—Sets the constraint as external and indicates that the policy set should apply to all external requests received through Oracle HTTP Server.

- `!HTTPHeader("VIRTUAL_HOST_TYPE","External")`—Sets the constraint as *NOT* external and indicates that the policy set should apply to all incoming requests not received through Oracle HTTP server, such as those from an internal network.

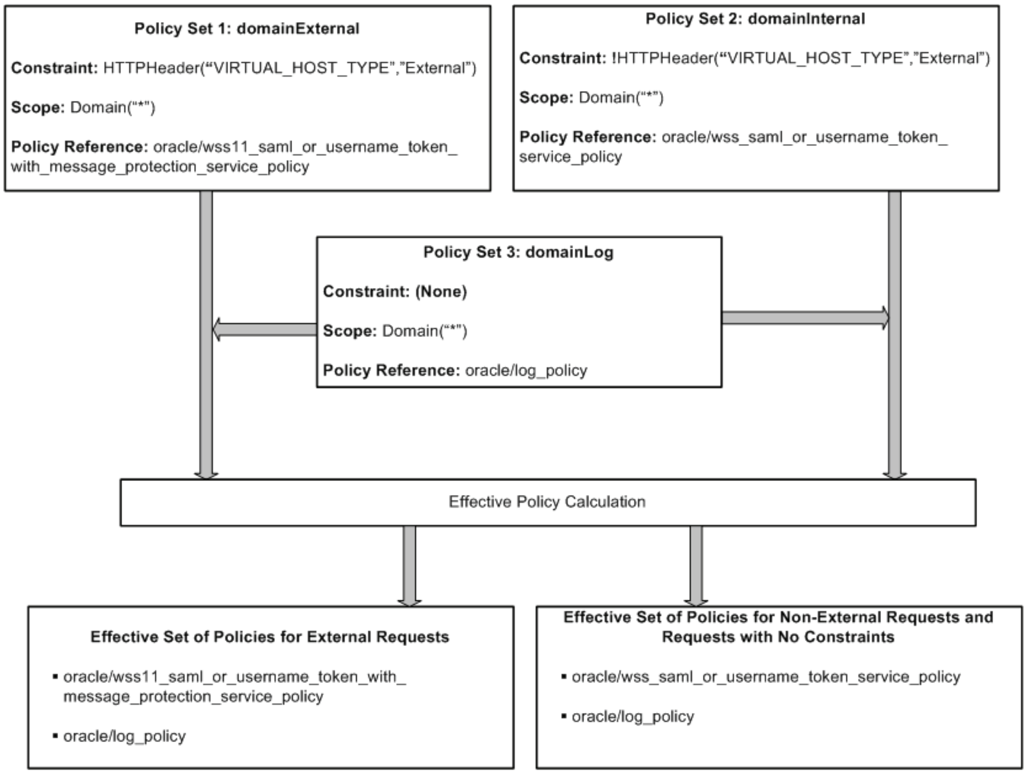
> **Note:** The run-time constraint function `HTTPHeader` is only certified to use when Oracle HTTP Server is configured at the front end *and* the Oracle HTTP Server administrator has added a custom `VIRTUAL_HOST_TYPE` header to the request. For details about adding the header to the request, see "Configuring Oracle HTTP Server to Specify Request Origin" on page 11-108. Although that procedure refers to the `oracle/whitelist_authorization_policy`, it also applies to specifying the request origin for run-time constraints.

When specifying constraints, the following rules apply:

- If multiple policy sets specify the same constraint, standard effective policy calculation rules apply. For details about the standard effective policy rules, see "How the Effective Set of Policies is Calculated" on page 9-38.
- If multiple policy sets specify different constraints, the effective set of policies is calculated against each type of constraint independently. That is, the effective set of policies is evaluated for all external requests and a separate effective set of policies is evaluated against all non-external requests.
- If no run-time constraint is specified in a policy set, it applies to all requests; that is, both external and non-external requests.

Figure 9–10 illustrates the effective policies for external and non-external requests determined using constraints in three different policy sets.

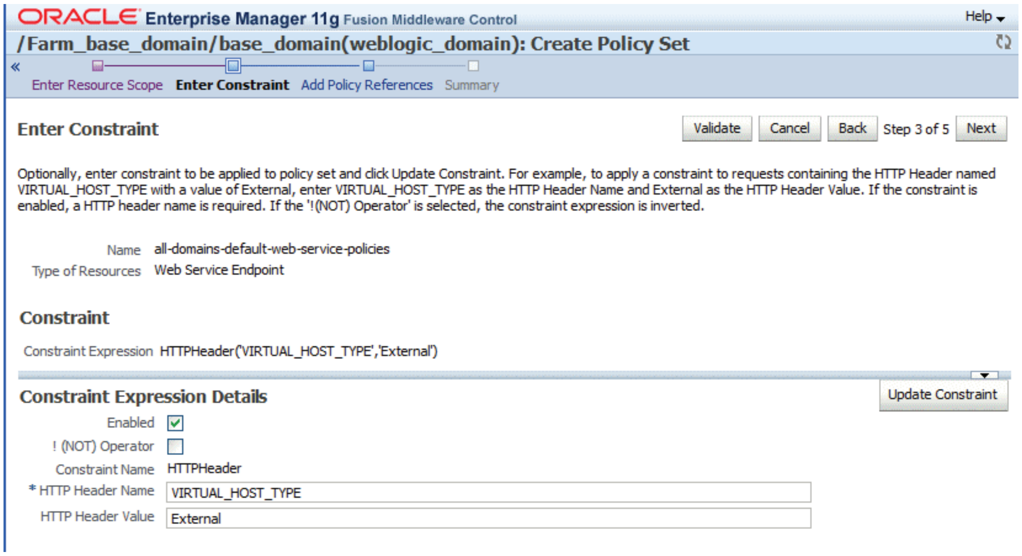
Figure 9–10 Effective Policy Calculation for Policy Sets with Run Time Constraints


Policy Set 1: domainExternal
Constraint: HTTPHeader(“VIRTUAL_HOST_TYPE”,”External”)
Scope: Domain(“*”)
Policy Reference: oracle/wss11_saml_or_username_token_with_message_protection_service_policy
Policy Set 2: domainInternal
Constraint: !HTTPHeader(“VIRTUAL_HOST_TYPE”,”External”)
Scope: Domain(“*”)
Policy Reference: oracle/wss_saml_or_username_token_service_policy
Policy Set 3: domainLog
Constraint: (None)
Scope: Domain(“*”)
Policy Reference: oracle/log_policy
Effective Policy Calculation
Effective Set of Policies for External Requests
▪ oracle/wss11_saml_or_username_token_with_message_protection_service_policy
▪ oracle/log_policy
Effective Set of Policies for Non-External Requests and Requests with No Constraints
▪ oracle/wss_saml_or_username_token_service_policy
▪ oracle/log_policy

9.12.1 Using Fusion Middleware Control

You can specify a run-time constraint when you are creating, editing, or cloning a policy set using the policy set wizard. After you specify the scope for the policy set, the Enter Constraint page is displayed. If you are editing or cloning a policy set with a constraint specified, the constraint currently configured in the policy set is displayed, as shown in Figure 9–11. If you are creating a new policy set, these fields are blank.

Figure 9–11 Enter Constraint Page in Policy Set Wizard


ORACLE Enterprise Manager 11g Fusion Middleware Control
Help
/Farm_base_domain/base_domain(weblogic_domain): Create Policy Set
Enter Resource Scope
Enter Constraint
Add Policy References
Summary
Validate
Cancel
Back
Step 3 of 5
Next
Update Constraint

Enter Constraint

Optionally, enter constraint to be applied to policy set and click Update Constraint. For example, to apply a constraint to requests containing the HTTP Header named VIRTUAL_HOST_TYPE with a value of External, enter VIRTUAL_HOST_TYPE as the HTTP Header Name and External as the HTTP Header Value. If the constraint is enabled, a HTTP header name is required. If the '!(NOT) Operator' is selected, the constraint expression is inverted.

Name all-domains-default-web-service-policies
Type of Resources Web Service Endpoint

Constraint

Constraint Expression HTTPHeader('VIRTUAL_HOST_TYPE','External')

Constraint Expression Details

Enabled ☑
! (NOT) Operator □
Constraint Name HTTPHeader
* HTTP Header Name VIRTUAL_HOST_TYPE
HTTP Header Value External

To specify a constraint:

1. In the Constraint Expression Details section of the page, select the **Enabled** check box to enable the constraint.
2. Optionally, select the **!(NOT) Operator** to invert the constraint.
3. Enter a header name and header value for the HTTPHeader constraint function in the **HTTP Header Name** and **HTTP Header Value** fields, respectively. If the constraint is enabled, the **HTTP Header Name** field is required.

   For example, as shown in Figure 9–11, to specify a constraint that applies to external clients only, enter `VIRTUAL_HOST_TYPE` in the **HTTP Header Name** field and `External` in the **HTTP Header Value** field.
4. Click **Update Constraint.**

   The constraint expression is displayed in the Constraint field, for example `HTTPHeader('VIRTUAL_HOST_TYPE','External')`.

9.12.2 Using WLST

You can specify a constraint in a policy set using the `setPolicySetConstraint` command. This command can be used only during the creation or modification of a policy set within the context of a repository session.

The following procedure describes how to specify a run-time constraint while creating a new policy set, but you can also use the `setPolicySetConstraint` command in a repository session while editing an existing policy set or creating a new policy set from an existing policy set.

1. Begin a repository session using the `beginRepositorySession` command.

For example:

```
wls:/jrfServer_domain/serverConfig> beginRepositorySession()

Repository session begun.
```

2. Use the `createPolicySet` command to create a new policy set.

   For example, to create a policy set for that provides authentication and message protection to external clients at the domain scope:

   ```
   wls:/jrfServer_domain/serverConfig>createPolicySet('domainExternal',
   'ws-service','Domain("*")','Authentication and message protection at domain
   scope for external clients')

   The policy set was created successfully in the session.
   ```

   For details about creating a policy set using WLST, see "Using WLST" in "Creating a Policy Set" on page 9-6.

3. Attach a policy to the current policy set using the `attachPolicySetPolicy` command.

   For example, to attach the policy `'oracle/wss10_message_protection_service_policy'` to the subjects specified in the policy set, enter the following command:

   ```
   wls:/jrfServer_domain/serverConfig>attachPolicySetPolicy('oracle/wss11_saml_or_
   username_token_with_message_protection_service_policy

   Policy reference added.
   ```

4. Specify a run-time constraint using the `setPolicySetConstraint(constraint)` command. The `constraint` argument must use a supported expression that defines a valid run-time constraint in a supported format. The following expressions are certified in this release:

   - `HTTPHeader("VIRTUAL_HOST_TYPE","External")`
   - `!HTTPHeader("VIRTUAL_HOST_TYPE","External")`

   For example, to specify a constraint that applies to external clients only, enter the following command:

   ```
   wls:/jrfServer_domain/serverConfig>setPolicySetConstraint('HTTPHeader("VIRTUAL_
   HOST_TYPE","External")')

   Constraint updated.
   ```

5. Optionally, display the configuration of the policy set during the current repository session using the `displayPolicySet` command.

   For example:

   ```
   wls:/jrfServer_domain/serverConfig>displayPolicySet()

   Policy Set Details:
   -------------------
   Name:                  domainExternal
   Type of Resources:     Web Service Endpoint
   Scope of Resources:    Domain("em_domain")
   Constraint:            HTTPHeader("VIRTUAL_HOST_TYPE","External")
   Description:           Authentication and message protection at domain scope for
   external clients
   Enabled:               true
   ```

```
Policy Reference:    security : oracle/wss11_saml_or_username_token_with_
message_protection_service_policy, enabled=true
```

6. Write the contents of the current repository session to the repository using the `commitRepositorySession` command.

```
wls:/jrfServer_domain/serverConfig> commitRepositorySession()

The policy set domainMsgExternal is valid.
Creating policy set domainMsgExternal in repository.
Repository session committed successfully.
```

9.13 Disabling a Globally Attached Policy

To explicitly disable a globally attached policy for specific endpoints, predefined policies that do not enforce any behavior are included with your Fusion Middleware installation. You can disable a globally, or externally, attached policy by attaching one of these predefined policies that contains the same category of assertions as the policy to be disabled. You can attach the no behavior policy either directly to an endpoint, or globally at a lower scope, such as at the application or module level. By default, a policy that is directly attached takes precedence over a policy that is globally attached and a policy that is globally attached at a lower scope takes precedence over a policy that is globally attached at a higher scope. For more information, see "How the Effective Set of Policies is Calculated" on page 9-38.

For example, if an authentication policy is globally attached to all service endpoints in a domain, you can disable it for a specific Web service endpoint by directly attaching the `oracle/no_authentication_service_policy` to the endpoint. Alternatively, to disable the authentication policy for only an application in the domain, you can create a policy set that attaches the `oracle/no_authentication_service_policy` only to the service endpoints in the application.

Note: If the globally attached policy that you are disabling contains any other assertions, those assertions are disabled also. For example, if the global policy to be disabled is `oracle/wss10_saml_token_with_message_protection_client_policy` and you attach the no behavior `oracle/no_authentication_service_policy` to an endpoint at lower scope (or directly), both the authentication and the message protection assertions of the globally attached policy are disabled.

For details about directly attaching a policy to an endpoint, see "Attaching a Policy to a Single Subject" on page 8-3. For more information about the no behavior policies, see "No Behavior Policies" on page B-41.

Note: Do not delete these no behavior policies. All of the policies use the same no_behavior assertion. An assertion template is not provided, therefore if you delete the policies, there is no way to recreate them manually. If they are deleted by mistake, the only way to restore them is to rebuild the repository. For more information, see "Rebuilding the Oracle WSM Repository" on page 17-7.

9.14 Enabling and Disabling a Policy Set

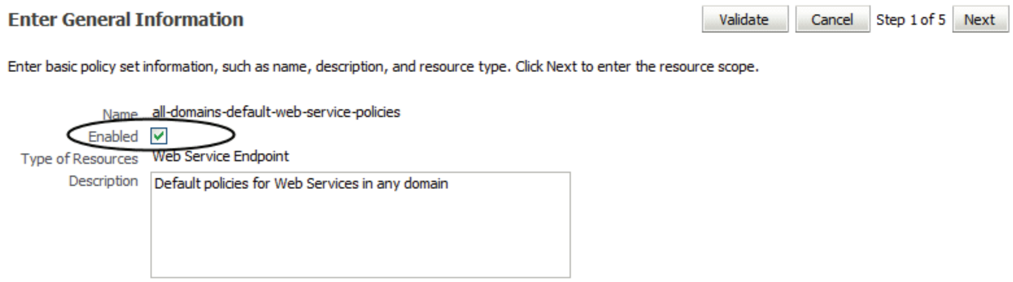
The following sections describe how to enable or disable a policy set using either Fusion Middleware Control or the command line interface WebLogic Scripting Tool, WLST.

9.14.1 Using Fusion Middleware Control

To enable or disable a policy set using Fusion Middleware Control, edit the policy set as described in "Editing a Policy Set" on page 9-15. To enable the policy set if it is disabled, select the **Enabled** check box. To disable the policy set, clear the **Enabled** check box.

Note that you must click **Next** through the remaining steps, then click **Save** to save the updated policy set.

Figure 9–12 Enabling and Disabling a Policy Set



Enter General Information

Enter basic policy set information, such as name, description, and resource type. Click Next to enter the resource scope.

Name all-domains-default-web-service-policies
Enabled ☑
Type of Resources Web Service Endpoint
Description Default policies for Web Services in any domain

9.14.2 Using WLST

To enable or disable a policy set:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Begin a repository session using the `beginRepositorySession` command.

   For example:

   ```
   wls:/jrfserver_domain/serverConfig> beginRepositorySession()

   Repository session begun.
   ```

3. Specify the policy set to be modified using the `modifyPolicySet` command.

   For example:

   ```
   wls:/jrfServer_domain/serverConfig>
   modifyPolicySet('all-domains-default-web-service-policies')

   The policy set is ready for modification in the session.
   ```

4. Use the `enablePolicySet` command to enable or disable a policy set.

   ```
   enablePolicySet([enable=true])
   ```

   Set the `enable` argument to `true` to enable a policy set if it is disabled. The default is `true`. Set the `enable` argument to `false` to disable a policy set.

   For example, to disable a policy set:

```
wls:/jrfServer_domain/serverConfig> enablePolicySet(false)

Policy set disabled.
```

5. Validate the policy set using the `ValidatePolicySet` command.

   For example:

   ```
   wls:/jrfServer_domain/serverConfig> validatePolicySet()

   The policy set app-only-web-service-policies is valid.
   ```

6. To write the contents of the current repository session to the repository, use the `commitRepositorySession` command.

   ```
   wls:/jrfServer_domain/serverConfig> commitRepositorySession()

   The policy set all-domains-default-web-service-policies is valid.
   Updating policy set all-domains-default-web-service-policies in repository.

   Repository session committed successfully.
   ```

   Alternately, you can choose to cancel any changes by using the `abortRepositorySession` command, which discards any changes that were made to the repository during the session.

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

9.15 Deleting Policy Sets

The following sections describe how to delete policy sets using Fusion Middleware Control or the command line interface WebLogic Scripting Tool, WLST.

9.15.1 Using Fusion Middleware Control

To delete a policy set:

1. Navigate to the Policy Set Summary page as described in "Navigating to the Policy Set Summary Page" on page 9-3.
2. In the Policy Set Summary page, select a policy set from the table and click **Delete**.
3. A dialog box displays asking you to confirm the deletion. Click **OK**.

9.15.2 Using WLST

You can use the following commands to delete policy sets in the repository:

- `deletePolicySet`—Deletes an individual policy set within the context of a repository session.
- `deleteAllPolicySets`—Delete select or all policy sets in the repository. This command can be used inside or outside a repository session.

To delete an individual policy set in a repository session:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Begin a repository session using the `beginRepositorySession` command.

   For example:

```
wls:/jrfserver_domain/serverConfig> beginRepositorySession()

Repository session begun.
```

3. Optionally, list the policy sets in the repository using the `listPolicySets` command.

   ```
   wls:/jrfServer_domain/serverConfig> listPolicySets()

   Global Policy Sets in Repository:
     app-only-web-service-policies
     all-domains-default-web-service-policies
   ```

4. Delete the desired policy set using the `deletePolicySet` command.

   ```
   deletePolicySet (name)
   ```

   For example:

   ```
   wls:/jrfServer_domain/serverConfig>
   deletePolicySet('app-only-web-service-policies')

   The policy set was deleted successfully in the session.
   ```

5. To write the contents of the current repository session to the repository, use the `commitRepositorySession` command.

   ```
   wls:/jrfServer_domain/serverConfig> commitRepositorySession()

   Deleting policy set app-only-web-service-policies from repository.

   Repository session committed successfully.
   ```

   Alternately, you can choose to cancel any changes by using the `abortRepositorySession` command, which discards any changes that were made to the repository during the session.

To delete all or select policy sets in the repository:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Optionally, list the policy sets in the repository using the `listPolicySets` command.

   ```
   wls:/jrfServer_domain/serverConfig> listPolicySets()

   Global Policy Sets in Repository:
     all-domains-default-web-service-policies
     ws-1
     ws-2
   ```

3. Delete the desired policy sets using the `deleteAllPolicySets()` command. You can specify whether to force deletion of all the policy sets (using the `force` argument), or prompt to select individual policy sets for deletion. This command defaults to `prompt` mode.

   ```
   deleteAllPolicySets(mode)
   ```

   For example, to specify the policy sets to be deleted:

   ```
   wls:/jrfServer_domain/serverConfig> deleteAllPolicySets()
   ```

```
Starting Operation deleteAllPolicySets ...
Policy Set Name: ws-2
Select "ws-2" for deletion (yes/no/cancel)? yes
Policy Set Name: all-domains-default-web-service-policies
Select "all-domains-default-web-service-policies" for deletion (yes/no/cancel)?
no
Policy Set Name: ws-1
Select "ws-1" for deletion (yes/no/cancel)? yes

All the selected policy sets were deleted successfully from repository.

deleteAllPolicySets Operation Completed.
```

To force the deletion of all policy sets:

```
wls:/jrfServer_domain/serverConfig> deleteAllPolicySets('force')

Starting Operation deleteAllPolicySets ...


All policy sets were deleted successfully from repository.

deleteAllPolicySets Operation Completed.
```

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

9.16 Migrating Direct Policy Attachments to Global Policy Attachments

You can use the `migrateAttachments` WLST command to migrate direct (local) policy attachments to external global policy attachments if they are identical. Migrating identical policy attachments improves manageability by reducing the number of physical attachments that need to be maintained.

A direct policy attachment is identical to a global policy attachment if its URI is the same as the URI provided by a global policy attachment, and if they both:

- Do *not* have any configuration overrides.

  or

- Do have scoped configuration overrides, and the direct policy attachment's scoped configuration override properties and values are the same as that of the global policy attachment.

You cannot migrate the following:

- Programmatic policy attachments.
- Direct or global policy attachments to SOA components

> **Notes:** The `migrateAttachments` WLST command does not have a way to identify unscoped overrides on direct policy attachments. Therefore, a direct policy attachment with an unscoped override will be treated as if it has no configuration overrides, and so it will be migrated if `migrateAttachments` finds an equivalent global policy attachment with no configuration overrides.

To migrate policy attachments:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Migrate the attachments using the `migrateAttachments` command. You can specify whether to force the migration (`force`), prompt for confirmation before each migration (`prompt`), or simply list the migrations that would occur (`preview`). If no mode is specified, the default is `prompt`.

   ```
   migrateAttachments(mode=None)
   ```

   For example, to prompt, by default, for confirmation of each potential attachment migration, enter the following command. Note in the output that there are identical global and direct policy attachments for the `jaxws-sut` application that can be migrated.

   ```
   wls:/jrfServer_domain/serverConfig> migrateAttachments()

   -------------------------------------------------------------------------------
   -
   Application Path:    /jrfServer_domain/jrfServer/jaxws-sut-no-policy
   Web Service Name:    TestService
   Module Type:         web
   Module Name:         jaxws-service
   Port:                TestPort

   -------------------------------------------------------------------------------
   -
   Application Path:    /jrfServer_domain/jrfServer/jaxws-sut
   Web Service Name:    TestService
   Module Type:         web
   Module Name:         jaxws-sut-service
   Port:                TestPort
   Policy Reference:    management : oracle/log_policy, enabled=true
                        security : oracle/wss_username_token_service_policy,
   enabled=true
                        (global) /policysets/global/migrate_example : oracle/wss_
   username_token_service_policy

   Migrate "oracle/wss_username_token_service_policy" (yes/no/cancel)? yes
   "oracle/wss_username_token_service_policy" was migrated successfully.
   ```

   For more information about the arguments for this command, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

9.17 Specifying the Priority of a Policy Attachment

The predefined policies provided in your installation include a configuration override, `reference.priority`, that allows an administrator to indicate a preference over which policy attachment is used. By default, an attached policy has a `reference.priority` of `0` if no other value has been specified.

For example, an administrator can globally attach a policy at the domain scope and specify a reference priority of 1 or greater to ensure that it takes precedence over any directly attached policies, without having to modify the direct attachments. If the administrator wants to make an exception for a particular direct attachment, then they can specify a reference priority for that attachment to elevate its priority above that of the global policy attachment. The policy attachment with the highest integer value for `reference.priority` takes precedence in the effective policy calculation, regardless of whether it is directly or externally attached, or its scope.

The value of `reference.priority` can be specified as follows:

- String values `"true"`, `"yes"` and `"on"`.

  These string values are equivalent to integer value 1. Any other string values will be treated as integer value 0.

- Integer values within the following range
  - MAX_VALUE = 2147483647 or (2^{31} - 1)
  - MIN_VALUE = -2147483648 or (-2^{31})

For more information, see the following topics:

- "Overriding Configuration Properties for Globally Attached Policies" on page 9-22
- "Attaching Web Service Policies Permitting Overrides" on page 8-25
- "How the Effective Set of Policies is Calculated" on page 9-38

9.18 Determining the Secure Status of an Endpoint

Global policy attachments provide the ability to adhere to a "secure by default" philosophy in which all subjects are secured even if the developer, assembler or deployer did not explicitly specify the policies to be attached. That is, using a policy set the administrator can ensure that one or more policies are automatically applied if none are explicitly attached.

An administrator can determine if all subjects in a domain are secure, and if the endpoint configuration is valid, using both WLST and Fusion Middleware Control.

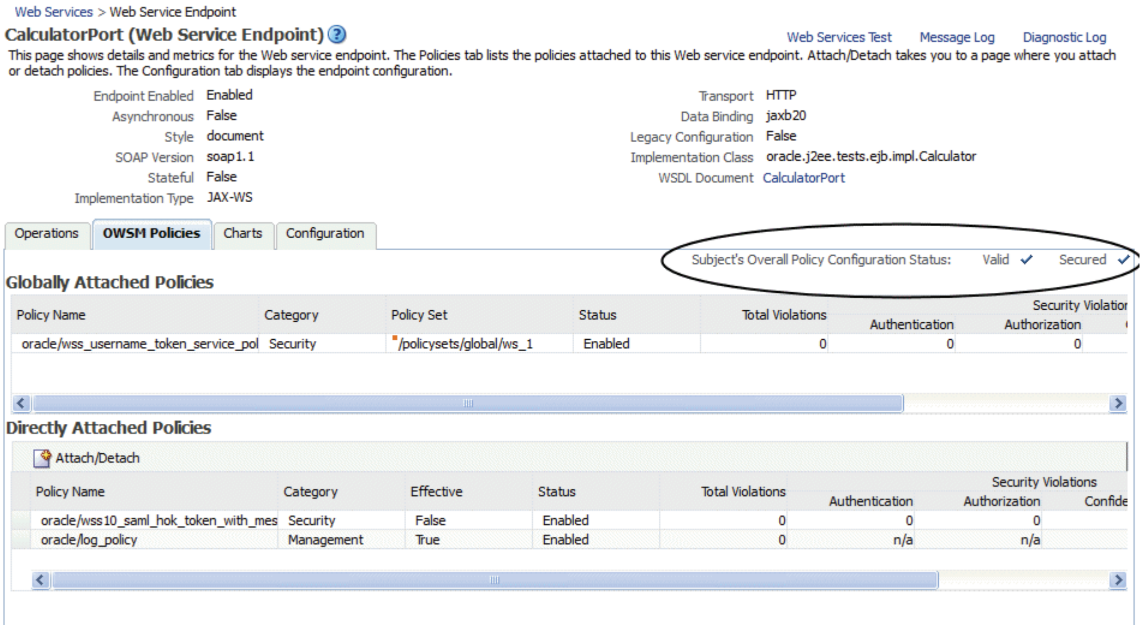
Note the following:

- An endpoint is considered secure if the policies attached to it (either directly or globally) enforce authentication, authorization, or message protection behaviors. A disabled policy or a disabled assertion within a policy does not enforce anything.
- An endpoint has a valid configuration if there is no conflict in the combination of attached policies according to the effective set of policies calculation. For more information, see "How the Effective Set of Policies is Calculated" on page 9-38

Because you can specify the priority of a globally or directly attached policy, as described in "Specifying the Priority of a Policy Attachment" on page 9-35, the Effective field for a directly attached policy indicates if it is in effect for the endpoint. Note that to simplify endpoint management, all directly attached policies are shown in the output regardless of whether they are in effect. In contrast, only globally attached policies that are in effect for the endpoint are displayed.

Using Fusion Middleware Control, you can view whether the configuration is valid and if the endpoint is secure on the Web Service Endpoint page. Figure 9–13 shows a valid configuration with a secure endpoint.

Figure 9–13 Web Service Endpoint Page With Valid and Secure Endpoint Configuration



Web Services > Web Service Endpoint

CalculatorPort (Web Service Endpoint)

Web Services Test   Message Log   Diagnostic Log

This page shows details and metrics for the Web service endpoint. The Policies tab lists the policies attached to this Web service endpoint. Attach/Detach takes you to a page where you attach or detach policies. The Configuration tab displays the endpoint configuration.

Endpoint Enabled	Enabled	Transport	HTTP
Asynchronous	False	Data Binding	jaxb20
Style	document	Legacy Configuration	False
SOAP Version	soap1.1	Implementation Class	oracle.j2ee.tests.ejb.impl.Calculator
Stateful	False	WSDL Document	CalculatorPort
Implementation Type	JAX-WS		

Subject's Overall Policy Configuration Status: Valid ✓ Secured ✓

Globally Attached Policies

Policy Name	Category	Policy Set	Status	Total Violations	Security Violatior	
					Authentication	Authorization
oracle/wss_username_token_service_pol	Security	/policysets/global/ws_1	Enabled	0	0	0

Directly Attached Policies

Policy Name	Category	Effective	Status	Total Violations	Security Violations		
					Authentication	Authorization	Confide
oracle/wss10_saml_hok_token_with_mes	Security	False	Enabled	0	0	0	
oracle/log_policy	Management	True	Enabled	0	n/a	n/a	

Using WLST, you can generate a list of endpoints and their secured status using the `listWebServices` and `listWebServiceClients` WLST commands. The output from these commands, when the detail argument is set to `true` as shown in Example 9–3, provides endpoint and policy details for all applications and composites in the domain, the secure status of the endpoints, any configuration overrides and constraints, and if the endpoints have a valid configuration.

Example 9–3 listWebServicesOutput with Valid and Secure Endpoint Configuration

```
wls:/jrfServer_domain/serverConfig> listWebServices(detail='true')

/jrfServer_domain/jrfServer_admin/jaxws-sut :
        moduleName=jaxws-sut-service, moduleType=web,
serviceName={http://namespace/}TestService
        enableTestPage: true
        enableWSDL: true


                TestPort
http://host.example.com:9315/jaxws-sut-service/TestService
                enable: true
                enableREST: false
                enableSOAP: true
                maxRequestSize: -1
                loggingLevel: NULL
                management : oracle/log_policy, enabled=true
                security : oracle/wss_username_token_service_policy , enabled=true
, effective=false
                Constraint: No Constraint
                        (global) security : oracle/wss_saml_or_username_token_
service_policy, enabled=true

/policysets/global/all-domains-default-web-service-policies : Domain("*")
                                        reference.priority=1
                Constraint: HTTPHeader('VIRTUAL_HOST_TYPE','external')
                        (global) security : oracle/wss10_message_protection_
service_policy, enabled=true
```

```
                                     /policysets/global/domainExternal : Domain("*")
                          Attached policy or policies are valid; endpoint is secure.
```

For more information about using these WLST commands, see "Viewing the Web Services in a Domain Using WLST" on page 6-2.

9.19 How the Effective Set of Policies is Calculated

Oracle WSM places a limit on the number of policies that may be attached to a subject based on the categories of the assertions that they contain. In most cases, attaching two or more policies containing the same assertion categories is forbidden. For example, it does not allow two policies containing authentication assertions to be attached to a policy subject, although it does allow one policy containing an authentication assertion and one containing an authorization assertion to be attached to the same subject. If multiple policies containing the same assertion category are attached to a subject, and the assertions conflict, the configuration is considered invalid. For details about the number and combination of policies that can be attached to a subject, see "Validating Policy Subjects" on page 8-10

To support the attachment of policies both directly and externally (globally), the determination of the effective set of policies for a subject takes into account the category of assertions within each policy. Note that policies that are directly attached are attached at the port or component scope. By default, if a subject has a policy attached at the port or component scope (such as a directly attached policy) with an assertion of a given category, then any policies with conflicting assertions of the same category referenced by an external policy set at a higher scope will be excluded from the effective set of policies for the subject, unless the `reference.priority` configuration override is set, as described below. This process will be repeated at each subject scope. Narrower/lower scopes take precedence over broader/higher scopes. The decreasing order of precedence of policy attachments as determined by scope is as follows:

- Port (Scope:1)
- Component (Scope:1)
- Service (Scope:2)
- Reference (Scope:2)
- Composite (Scope: 3)
- Module (Scope:3
- Partition (Scope:4)
- Application (Scope:4)
- Server (Scope:5)
- Domain (Scope:6)

For example, a policy attachment at the application scope will be excluded from the effective set of policies for a subject if it contains conflicting assertions of the same category as a policy that was attached at the module scope or attached directly. For additional information about resource scopes, see "Defining the Type and Scope of Resources" on page 9-18.

By using the `reference.priority` configuration override, the administrator can override the default precedence determined by scope and specify a preference over which policy attachment is used. The policy attachment with the highest priority takes precedence, irrespective of its scope.

When using `reference.priority` overrides, the following rules apply:

- The policy attachment with the highest priority (highest integer value) takes precedence, regardless of scope.
- If attachments contain conflicting assertions of the same category and have the same priority specified, the more specific scope takes precedence.
- If attachments contain conflicting assertions of the same category, priority, and scope, then the configuration is invalid.

When run-time constraints are applied to policy sets, the following rules apply:

- Each unique constraint creates an independent set of policies. The effective policy calculation is performed only on the set of policies with the same constraint.
- When no constraint is specified in a policy set (the default), the policy reference in this set is merged with the set of policies from each separate constraint. The effective policy calculation is then performed on each set of policies to determine the effective set of policies for each constraint.

For more information about run-time constraints, see "Specifying Run-time Constraints in Policy Sets" on page 9-26.

The effective set of policies calculation takes into account the status of each policy attachment. If a policy, a policy reference in a policy set, or a policy set is disabled, it is removed from the effective set of policies for a subject.

If *no* `reference.priority` override is specified, a globally attached policy can be overridden by attaching a policy containing assertions with the same categories at a lower scope (for example at the port or component scope with a direct attachment). As a special case of this, a globally attached policy can be effectively disabled for a specific subject by attaching a policy with the same category of assertions that does not enforce any behavior. For more information about the policies that do not enforce any behavior, see "No Behavior Policies" on page B-41.

The following examples demonstrate the results in effective policy calculations:

- Direct attachment: `oracle/wss_username_token_service_policy`

  External attachment: `oracle/wss_saml_or_username_token_service_policy @ Domain('*')`

  Result: Direct attachment due to lower scope— `oracle/wss_username_token_service_policy`

- Direct attachment: `oracle/wss_username_token_service_policy`

  External attachment: `oracle/wss_saml_or_username_token_service_policy @ Domain('*')` with `reference.priority=1`

  Result: External attachment due to higher priority—`oracle/wss_saml_or_username_token_service_policy @ Domain('*')`

- External attachment: `oracle/wss_username_token_service_policy @ Application('*')`

  External attachment: `oracle/wss_saml_or_username_token_service_policy @ Domain('*')`

  Result: External attachment due to lower scope—`oracle/wss_username_token_service_policy @ Application('*')`

- External attachment: `oracle/wss_username_token_service_policy @ Application('*')`

External attachment: `oracle/wss10_message_protection_service_policy @ Domain('*')`

Result: Both attachments valid due to non-conflicting assertion categories—`oracle/wss_username_token_service_policy @ Application('*')` and `oracle/wss10_message_protection_service_policy @ Domain('*')`

- External attachment: `oracle/wss_username_token_service_policy @ Domain('*')`

  External attachment: `oracle/wss_saml_or_username_token_service_policy @ Domain('*')`

  Result: Invalid. Policies with conflicting assertion categories specified at the same scope.

- Direct attachment: `oracle/wss11_saml_token_with_message_protection_service_policy`

  External attachment: `oracle/wss11_username_token_with_message_protection_service_policy @ Port('TestPort')`

  Result: Invalid. Policies with conflicting assertion categories specified at the same scope.

- Direct attachment: `oracle/wss11_saml_token_with_message_protection_service_policy`

  External attachment: `oracle/wss11_username_token_with_message_protection_service_policy @ Port('TestPort')` with `reference.priority="true"`

  Result: External attachment due to higher priority scope—`oracle/wss11_username_token_with_message_protection_service_policy @ Port('TestPort')`

Note: The amount of time it takes for a global policy attachment to take effect is determined by the Oracle WSM policy accessor and policy cache property settings. By default, this delay can be up to a maximum of 11 minutes. To reduce the amount of the delay, you can tune the following cache property settings:

- Policy Accessor

  `cache.refresh.initial`, default 600000 milliseconds (10 minutes)

  `cache.refresh.repeat`, default 600000 milliseconds (10 minutes)

- Policy Cache

  `cache.tolerance`, default is 60000 milliseconds (1 minute)

For details about tuning these properties, see "Configuring Platform Policy Properties" on page 14-15.

10

Setting Up Your Environment for Policies

This chapter describes how to set up your Fusion Middleware Control and WebLogic Server environments for security policies.

This chapter includes the following sections:

- Understanding Keys and Certificates
- Configuring Keystores for Message Protection
- Configuring the Credential Store
- Creating an Application-level Credential Map
- Configuring the OPSS Keystore Service for Message Protection
- Configuring Keystores for SSL
- Configuring SSL on Oracle HTTP Server
- Hardware Integration
- Using Service Identity Certification Extension
- Configuring an Authentication Provider in WebLogic Server
- Configuring the SAML and Kerberos Login Modules
- Configuring SAML
- Using JSON Web Token (JWT) with Oracle WSM
- Using OAuth2 with Oracle WSM
- Configuring Web Service Clients for Identity Switching
- Propagating Identity Context with Oracle WSM
- Using Kerberos Tokens
- Using Active Directory with Kerberos and Message Protection
- SAML Message Protection Use Case
- WS-Trust Policies and Configuration Steps
- Examples Using WS-Trust with OpenSSO STS
- Understanding Fine-Grained Authorization Using Oracle Entitlements Server

10.1 Understanding Keys and Certificates

Before you can use any message protection security policies or message protection and authentication with SSL security policies, you need to set up your keystores and truststores. (Authentication-only security policies do not require keys.)

The keystore contains the entities private keys and certificates associated with those private keys. A truststore contains certificates from a Certificate Authority (CA), or other entities that this entity trusts. The keystore and the truststore can be maintained together in a common store, such as with Oracle Web Services Manager (WSM).

Before configuring your Web services, you need to determine the type of private keys and certificates required, the names for the keys and keystores, and then set up your environment accordingly.

10.1.1 Overview of Private Keys and Certificates

Private keys, digital certificates, and trusted certificate authorities establish and verify server identity and trust.

SSL uses public key encryption technology for authentication. With public key encryption, a public key and a private key are generated for a server. Data encrypted with the public key can only be decrypted using the corresponding private key and data verified with a public key can only have been signed with the corresponding private key. The private key is carefully protected so that only the owner can decrypt messages that were encrypted using the public key.

The public key is embedded in a digital certificate with additional information describing the owner of the public key, such as name, street address, and e-mail address. A private key and digital certificate provide identity for the server.

The data embedded in a digital certificate is verified by a certificate authority and digitally signed with the certificate authority's digital certificate. Well-known certificate authorities include Verisign and Entrust.net. The trusted certificate authority (CA) certificate establishes trust for a certificate.

An application participating in an SSL connection is authenticated when the other party evaluates and accepts the application's digital certificate. Web browsers, servers, and other SSL-enabled applications generally accept as genuine any digital certificate that is signed by a trusted certificate authority and is otherwise valid. For example, a digital certificate can be invalidated because it has expired or the digital certificate of the certificate authority used to sign it expired. A server certificate can be invalidated if the host name in the digital certificate of the server does not match the URL specified by the client.

The different types of trusted certificates, along with the benefits and disadvantages of each, that you can use in your environment are as follows:

- **Self-signed certificates** — A self-signed certificate is a certificate that is signed by the entity creating it.

  Benefits:

  - Easy to generate because you can do it yourself, for example using the keytool command as described in "Generating Private Keys and Creating the Java Keystore" on page 10-9.
  - Can be used in production as long as you use only a new certificate that you have generated.

  Disadvantages:

  - Self-signed certificates can quickly become unmanageable if you have many clients and services that need to communicate with each other. For example, if you have three clients communicating with two services, you need to generate a private key and self-signed certificate for both services, and then import the two certificates into the truststore of all three clients.

- **Demonstration Certificate Authority (CA) signed certificates**— WebLogic Server includes a set of demonstration private keys, digital certificates, and trusted certificate authorities that are for development only.

  Benefits:

  - Easy to use because they are available and configured for use in the default WebLogic Server installation in a development environment.

  Disadvantages:

  - Should never be used in a production environment. The private key of the demo certificate CA is available to all installations of WebLogic Server, therefore each installation can generate a demo CA signed certificate using the same key. As a result, you cannot trust these certificates.

- **Internal CA signed certificates** — An internal CA signed certificate is a certificate that you issue yourself using an internal CA that you can setup for your intranet. This type of certificate can be used if your services are mostly internal only.

  Benefits:

  - You have complete control over the certificate issuance process because you create the certificates yourself.You can control to whom the certificates are issued, how long the certificates remain valid, and so on. For example, if you are issuing certificates to your partners, you can issue them only to partners in good standing.

  Disadvantages:

  - You need to ensure that all clients have the internal CA root certificate imported into their truststore.

- **External CA signed certificates** — An external CA signed certificate is a certificate that has been issued by a reputable CA such as Verisign and Entrust.net. This type of certificate should be used if your services are external facing.

  Benefits:

  - In most cases, clients are already set up to trust these external CAs. Therefore, those clients do not have to modify their truststore.

  Disadvantages:

  - You do not have any control over the certificate issuance process.

10.1.2 How Different Security Policies Use Private Keys and Certificates

Oracle WSM security policies that require the use of private keys address two aspects: message protection and authentication:

- Message protection encompasses two concepts, **message confidentiality** and **message integrity**. Message confidentiality involves keeping the data secret and is achieved by encrypting the content of messages. Message integrity ensures that a message remains unaltered during transit by having the sender digitally sign the message.

- Authentication involves verifying that the user is who they claim to be. A user's identity is verified based on the credentials presented by that user.

The predefined Oracle WSM policies that are included with your installation support various options for message protection and authentication. These options are described in the following sections.

> **Note:** The naming convention used for Oracle WSM policies identifies the type of options being used. For example, the policy `oracle/wss10_username_token_with_message_protection_service_policy` is a message protection service policy that uses the wss10 Web services standard and requires a username_token for authentication. For more information about policy naming conventions, see "Recommended Naming Conventions for Policies" on page 3-10.

10.1.2.1 Message Protection Policy Types

The types of message protection policies and how they work are described in the following sections.

10.1.2.1.1 SSL Policies that include the SSL option, such as `oracle/wss_saml_or_username_token_over_ssl_service_policy`, use one-way SSL for message protection. When using policies of this type, you need to do the following:

- On the service side, set up private keys at the SSL termination point as described in "Setting Up Private Keys and Certificates for SSL Policies" on page 10-7.
- On the client side, set up the truststore to trust the service keys.

The private key is used to protect the messages for the SSL handshake, at which time the client and service agree on a shared session key. After the SSL handshake, the private key is not used, and all traffic between the client and the service are signed and encrypted using the shared session key.

10.1.2.1.2 wss11 Policies of this type use WS-Security 1.1 for message protection. When using wss11 policies, you need to do the following:

- On the service side, set up private keys and define as the Encryption Key Alias in the Oracle WSM Keystore Configuration screen. For details see "Configuring the Oracle WSM Keystore" on page 10-11.
- On the client side, you need to configure the client-side trust by obtaining the server's certificate in one of the following ways:
  - Use the service's public certificate published in the WSDL using the Service Identity Certificate extension as described in "Using Service Identity Certification Extension" on page 10-57. You also need to import either the server certificate itself, or the root certificate from the CA that issued the server certificate, into the client truststore. You can choose any alias name for the server certificate.
  - Import the server certificate into the client keystore using any alias you choose, and specify that alias using the `keystore.recipient.alias` property using a configuration override when you attach the policy. For this method you need to import the actual server certificate, you cannot import the CA root certificate.

For each request, the following occurs:

1. The client creates a symmetric key, encrypts this symmetric key with the service's public key as configured with Encryption Key Alias, and then encrypts and signs the whole message with the symmetric key.
2. When the service receives the message, it decrypts the encrypted key first, and then decrypts and verifies the whole message.
3. The Web service then uses the same symmetric key to encrypt and sign the response that it sends back to the client.

10.1.2.1.3 wss10 Policies of this type use WS-Security 1.0 for message protection. When using wss10 policies, you need to do the following:

- Set up private keys on both the client and service side. On the client side, you need to set a signature key alias, and on the service side you need both an encryption key alias and signature key alias. Note that you can normally use the same key for both.
- On the client side, you need to configure the client-side trust by obtaining the server's certificate in one of the following ways:
  - Use the service's public certificate published in the WSDL using the Service Identity Certificate extension as described in "Using Service Identity Certification Extension" on page 10-57. You also need to import either the server certificate itself, or the root certificate from the CA that issued the server certificate, into the client truststore. You can choose any alias name for the server certificate.
  - Import the server certificate into the client keystore using any alias you choose, and specify that alias using the `keystore.recipient.alias` property using a configuration override when you attach the policy. For this method you need to import the actual server certificate, you cannot import the CA root certificate.
- On the service side, you need to configure the service to trust the client, either by importing these certificates directly, or importing the CA that issued these certificates.

Similar to the wss11 option, the client creates a symmetric key, and then encrypts the symmetric key with the service's public key. The difference, however, is that it only uses this symmetric key for encrypting the message; it doesn't use it for signing the message. Instead, the client signs the request message with its own private signature key as defined by the Signature Key alias, and the service signs the response with its private signature key.

10.1.2.2 Authentication Token Policy Types

The tokens that are supported for authentication, and the private keys and certificates that are used with these policy types are described in the following sections.

Note that in the following sections, "signature key alias" is used to mean different things in different contexts.

- In SAML sender vouches policies, it is the key used to sign the SAML assertion. This proves the authenticity of the SAML assertion, and SAML Login module will then assert the user specified in the SAML assertion.
- In wss10 policies, it is used to sign the request and response message to prevent them from being tampered over the wire.
- In X.509 authentication policies, it is used to authenticate a particular end user.

10.1.2.2.1 JWT Token JSON Web Token (JWT) is a compact URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is digitally signed using a JSON Web Signature (JWS) and optionally encrypted using JSON Web Encryption (JWE). The token is generated and signed by the client using its private signature key.

10.1.2.2.2 Kerberos Token A Kerberos token is comprised of a binary authentication and session token. When a Kerberos token is used with an authentication-only policy, no private keys are used. When used in a policy that includes authentication and message protection, the keys required for message protection are required.

10.1.2.2.3 SAML Bearer and SAML HOK Tokens from an STS For these options, the client does not construct the SAML token. Instead it is STS that constructs and signs the SAML token.

When using tokens from an STS, you must add the STS's certificate or its issuer to the service's truststore. Optionally, you can configure the STS in the Trusted DN list.

10.1.2.2.4 SAML Sender Vouches Token In sender vouches, the client signs the SAML token with its own private signature key.

Use the SAML sender vouches token with each of the message protection options as follows:

- With SSL: SAML sender vouches requires two-way SSL. Therefore, you need to set up an SSL client-side private key, and corresponding trust certificate on the service side. If your SSL terminates before WebLogic Server, such as in the Oracle HTTP Server or in the Load balancer, you must configure these layers to propagate the client certificate all the way to WebLogic Server.
- With wss11: Normally wss11 does not need a client-side signature key. However, when you use wss11 with SAML, you need to set up a signature key on the client side, and configure it using the signature key alias. You also need to add this client certificate or its issuer to the service's truststore.
- With wss10: There is no additional setup to use SAML. The regular client signature key that is used for signing the request is also used for signing the SAML token.

Note: Be very cautious when using the SAML signature key. It is a very powerful key as it enables the client side to impersonate any user. Consider configuring the server side to limit the number of SAML signers that is accepts, by setting up a Trusted DN list. For information about setting up a trusted DN, see "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23.

10.1.2.2.5 Username Token A username token carries basic authentication information such as a username and password. When a username token is used with an authentication-only policy, no private keys are used. When used in a policy that includes authentication and message protection, the keys required for message protection are required.

10.1.2.2.6 X.509 Certificate Token Request messages are signed with the end user's signature key. On the client side you need to configure a signature key alias with the end user's signature key.

10.1.3 Setting Up Private Keys and Certificates for SSL Policies

The following list summarizes the keys and trust you need to configure on the client and service side to use SSL policies:

- **Service-side configuration:** For SSL security policies, you need to setup the private keys at the SSL termination point. These termination points typically consist of one of the following:
  - Java EE container, such as WebLogic Server. For configuration details, see "Configuring Keystores for SSL" on page 10-36.
  - Oracle HTTP Server, if you have configured it as a Web proxy between the client and WebLogic Server. For configuration details, see "Configuring SSL on Oracle HTTP Server" on page 10-43.
  - Load balancer, if you have a load balancer in front of WebLogic Server or Oracle HTTP Server.

  > **Note:** With SSL you can only have one private key per server, so if there are multiple Web services running on the same server, they all use the same private key. This SSL private key needs to be generated with the same DN as the host name, although for testing purposes, you can turn off the host name verifier on the client side.

  Sample basic configuration: Use the demonstration digital certificates, private keys, and trusted CA certificates that are included with WebLogic Server. These keys and certificates are provided for development use only and should not be used in a production environment.

  Advanced configuration: In a production environment, use an internal or external CA.

- **Client-side configuration:** On the client side, you need to import the server certificates into the client truststore. If the server side is using self-signed certificates, you need to include them directly. If the server side is using certificates that are signed using a CA, import the CA root certificate into the client truststore. Note that each type of Web service client has a different client truststore:
  - For WebLogic Server Web services, you need to import the keys into the WebLogic Server trust store. The demonstration CA certificate is already present in the WebLogic Server truststore.
  - For Oracle Infrastructure Web services you need to specify the truststore using javax.net.ssl* system properties, or specify it in the connection. For details, see "Configuring SSL for a Web Service Client" on page 10-41.
  - For SOA composite applications, you need to specify the truststore using the javax.net.ssl* property as described in "Configuring SOA Composite Applications for Two-Way SSL Communication" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.
  - For asynchronous Web services, you need to configure the truststore as described in "Configuring SSL for Asynchronous Web Services" in *Developer's Guide for Oracle Infrastructure Web Services*.

10.1.4 Setting up Private Keys and Certificates for Message Protection Policies

For Oracle WSM message protection security policies, you need to setup your private keys in the Oracle WSM keystore.

There is a single Oracle WSM keystore per domain, and it is shared by all Web services and clients running in the domain. This keystore contains both private keys and trust certificates. The JDK cacerts file is not used by Oracle WSM.

Sample Basic Configuration

The easiest way to set up the Oracle WSM keystore is to create a single self-signed private key and use it for the entire domain. When you create the private key and keystore, you specify a name and a password for the keystore, for example `default-keystore.jks` as the keystore name and *`password`* as the password for the keystore. You also specify an alias name and password to use when referring to the private key, for example `orakey` as the alias name and *`password`* as the key password. You can use the same key and alias for both the signature key alias and the encryption key alias, and the same password for both the keystore and the alias. You do not need to add any trusted certificates, as certificates associated with private keys are automatically considered as trusted.

Once you have created the keys and keystore, you need to provide the keystore password, and alias names and passwords to Oracle Web Services Manager.You can do so using either Fusion Middleware Control or WLST.

The procedures in "Generating Private Keys and Creating the Java Keystore" on page 10-9 and "Configuring the Oracle WSM Keystore" on page 10-11 describe how to setup this basic configuration using the names and passwords specified in this example. In your own environment, you should use names and passwords that are appropriate for your configuration.

As long as your client and server are on the same domain, this set up is sufficient to work with most of the policies. That is, you can use any wss10 or wss11 policies with or without SAML.

If you have multiple related domains that share a common JPS root, you can copy this keystore file to all the domains. By doing so, all the related domains will share this single key for all encryption and signing.

Advanced Setup Considerations

As described in "Sample Basic Configuration" on page 10-8, the simplest way to set up message protection security is to have a single private key for all Web services in the domain.

For more sensitive Web services, you need to configure each Web service to use its own distinct private encryption key. These private keys need to exist in the Oracle WSM keystore. Ensure that each one uses a different alias name, for example `ServiceA`, and `ServiceB`, and that you add the aliases to the credential store. When you attach a policy to the service, you need to use a configuration override to indicate the specific alias name that the Web service requires, otherwise it will use the default alias that you configured for the domain, for example `orakey`.

The procedure in "Adding Keys and User Credentials to the Credential Store" on page 10-19 describes how to add these sample aliases to the credential store.

You should also use trusted certificates issued by an internal or external CA, instead of self-signed certificates, because it is much easier to manage the trusted CA certificates. Be sure, however, to set up the SAML signers Trusted DN list, as described in "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23. This is especially important if you import external CA certificates into the Oracle WSM Keystore, otherwise any user with a certificate will be able to sign a SAML token and impersonate any user.

10.2 Configuring Keystores for Message Protection

> **Note:** To configure keystores for message protection using the REST API, see "JKS Keystore Management" in *REST API for Managing Credentials and Keystores with Oracle Web Services Manager*.

Message protection involves encrypting the message for message confidentiality and signing the message for message integrity. To sign and encrypt SOAP messages, you use public and private signature and encryption keys that you store in the Oracle Web Services Manager (WSM) keystore for the WebLogic domain. The keystore configuration is domain wide: all Web services and Web service clients in the domain use this keystore.

> **Note:** The Oracle WSM run time does not use the WebLogic Server keystore that is configured using the WebLogic Server Administration Console and used for SSL as documented in "Configuring Keystores for SSL" on page 10-36.

To create and configure the Java Keystore for message protection, use the procedures in the following sections:

- Generating Private Keys and Creating the Java Keystore
- Configuring the Oracle WSM Keystore

> **Note:** These procedures describe how to setup the basic configuration, using the names and passwords specified in the example, as described in "Sample Basic Configuration" on page 10-8, and illustrated in Figure 10–8. In your own environment, you should use names and passwords that are appropriate for your configuration.

10.2.1 Generating Private Keys and Creating the Java Keystore

The following section provides an outline of how to create a private key pair and the Java keystore (JKS) using the keytool utility. You can find more detailed information on the commands and arguments for the keytool utility at this Web address.

`http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html`

1. Go to the *`domain_home`*`/config/fmwconfig` directory, where *`domain_home`* is the name and location of the domain for which the keystore is to be used.
2. Enter a keytool command such as the following to generate the key pair, and to create the keystore if it does not already exist:

   ```
   keytool -genkeypair -keyalg RSA -alias orakey -keypass password
   -keystore default-keystore.jks -storepass password -validity 3600
   ```

> **Note:** You may need to add the `jdk/bin` directory to your PATH variable definition to invoke the `keytool` command.

In this command:

- `genkeypair` creates a new public/private key pair that is stored in an entry specified by the `alias` parameter
- `keyalg` specifies the algorithm to be used to generate the key pair, in this example `RSA`

> **Note:** The default key pair generation algorithm is Digital Signature Algorithm (DSA). DSA keys can only be used for signing, whereas RSA keys can be used for both signing and encryption. Therefore, if you are using the same key for encryption and signing (which is a typical scenario), make sure you explicitly specify `-keyalg RSA`, otherwise keytool will default to DSA.

- `alias` specifies the alias name `orakey` to use when referring to the keypair
- `keypass` specifies that the password be used to protect the private key of the generated key pair
- `keystore` creates a keystore named `default-keystore.jks`. If the keystore already exists, the key pair will be added to the keystore.
- `storepass` specifies `password` as the password used to protect the integrity of the keystore.
- `validity` indicates that the keypair is valid for 3600 days.

The keytool utility prompts for the name, organizational unit and organization, locality (city, state, country) to be used to create the key:

```
What is your first and last name?
  [Unknown]:  orcladmin
What is the name of your organizational unit?
  [Unknown]:  Doc
What is the name of your organization?
  [Unknown]:  Oracle
What is the name of your City or Locality?
  [Unknown]:  US
What is the name of your State or Province?
  [Unknown]:  US
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=orcladmin, OU=Doc, O=Oracle, L=US, ST=US, C=US correct?
  [no]:  y
```

3. Optionally, import trusted certificates into the keystore as described in "Obtaining a Trusted Certificate and Importing it into the Keystore" on page 10-15

4. Optionally, use the `keytool -list` command to view the contents of the keystore:

   ```
   keytool -list -keystore default-keystore.jks
   ```

   When prompted, provide the password for the keystore that you specified when you created the keystore.

   ```
   Enter keystore password:
   ```

```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: orakey
Creation date: Mar 9, 2011
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=orcladmin, OU=Doc, O=Oracle, L=US, ST=US, C=US
Issuer: CN=orcladmin, OU=Doc, O=Oracle, L=US, ST=US, C=US
Serial number: 4d77aff6
Valid from: Wed Mar 09 11:51:02 EST 2011 until: Fri Jan 15 11:51:02 EST 2021
Certificate fingerprints:
         MD5:  DF:EC:3C:60:CF:8B:10:A7:73:3A:51:99:4C:A3:D0:2E
         SHA1: E0:52:58:EB:34:51:E4:9B:D4:13:C2:CB:F3:CC:08:89:EF:4E:4E:05
         Signature algorithm name: SHA1withRSA
         Version: 3


*******************************************
*******************************************
```

10.2.2 Configuring the Oracle WSM Keystore

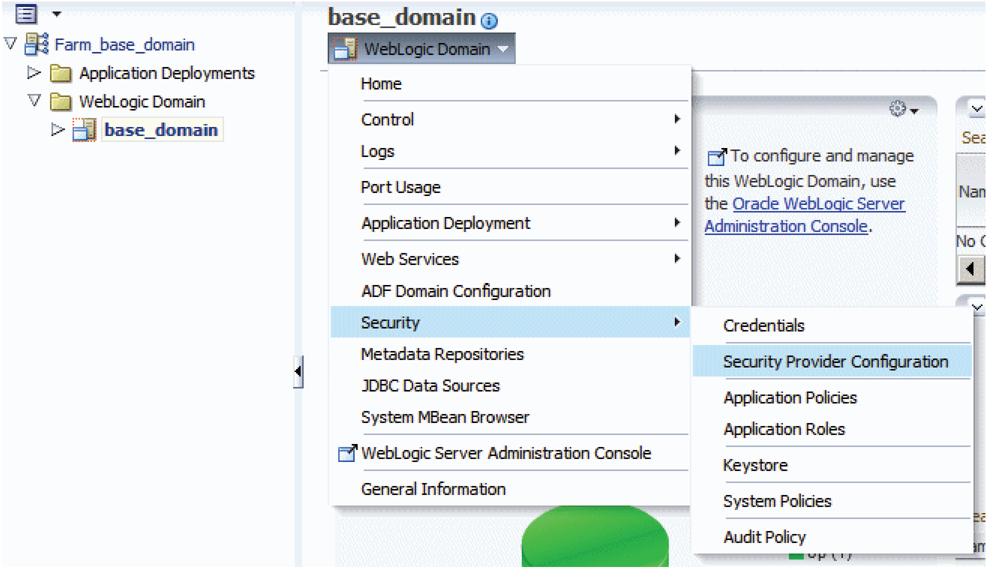
The following section describes how to use Oracle Enterprise Manager Fusion Middleware Control to configure the Oracle WSM keystore. This is the recommended method for configuring the keystore. If your environment does not include Fusion Middleware Control, you can also use WebLogic Scripting Tool (WLST) commands, as described in "Using WLST" on page 10-14.

10.2.2.1 Using Fusion Middleware Control

When you use Fusion Middleware Control to configure the Oracle WSM keystore, entries are created in the credential store for the credential map `oracle.wsm.security`, and any keys that you define. Use the following procedure to configure the keystore:

1. In the Navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the keystore. Select the domain.
2. From the **WebLogic Domain** menu, select **Security** then **Security Provider Configuration**, as shown in Figure 10–1.

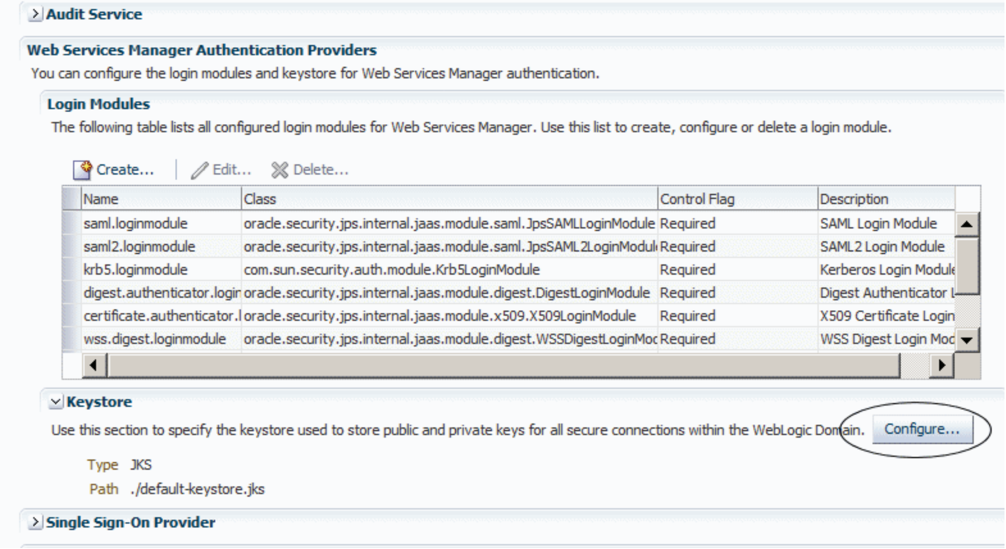
Figure 10–1 Security Provider Configuration Menu



To configure and manage this WebLogic Domain, use the Oracle WebLogic Server Administration Console.

The Security Provider Configuration page is displayed, as shown in Figure 10–2.

Figure 10–2 Security Provider Configuration Page



Web Services Manager Authentication Providers

You can configure the login modules and keystore for Web Services Manager authentication.

Login Modules

The following table lists all configured login modules for Web Services Manager. Use this list to create, configure or delete a login module.

Name	Class	Control Flag	Description
saml.loginmodule	oracle.security.jps.internal.jaas.module.saml.JpsSAMLLoginModule	Required	SAML Login Module
saml2.loginmodule	oracle.security.jps.internal.jaas.module.saml.JpsSAML2LoginModul	Required	SAML2 Login Module
krb5.loginmodule	com.sun.security.auth.module.Krb5LoginModule	Required	Kerberos Login Modul
digest.authenticator.login	oracle.security.jps.internal.jaas.module.digest.DigestLoginModule	Required	Digest Authenticator L
certificate.authenticator.l	oracle.security.jps.internal.jaas.module.x509.X509LoginModule	Required	X509 Certificate Login
wss.digest.loginmodule	oracle.security.jps.internal.jaas.module.digest.WSSDigestLoginMoc	Required	WSS Digest Login Moc

Keystore

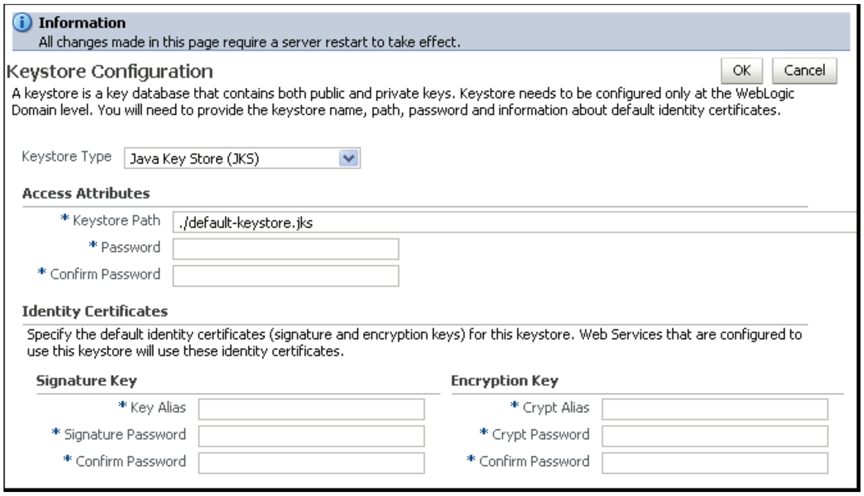
Use this section to specify the keystore used to store public and private keys for all secure connections within the WebLogic Domain.

Type JKS
Path ./default-keystore.jks

3. Click **Configure** in the Keystore section of the page.

   The Keystore Configuration page is displayed, as shown in Figure 10–3.

Figure 10–3 Keystore Configuration



Information

All changes made in this page require a server restart to take effect.

Keystore Configuration

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. You will need to provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Java Key Store (JKS)

Access Attributes

* Keystore Path: ./default-keystore.jks

Identity Certificates

Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore will use these identity certificates.

4. In the **Keystore Type** drop-down, select **Java Key Store (JKS)**, if it is not already selected.

> **Notes:** Hardware security modules (HSM) are also certified to operate with Oracle Advanced Security. For more information, see "Using Hardware Security Modules With Oracle WSM" on page 10-47
>
> You can also select **Public Key Cryptographic Standards (PKCS-11)** when using cryptographic acceleration, as described in Step 6 of "Configuring Message-level Security for Cryptographic Acceleration" on page 10-54. You use the PKCS-11 keystore type when hardware token support is required.

5. In the Access Attributes section of the page, provide the name and path of the keystore, and the passwords as follows:

   - In the **Keystore Path** field, enter the path and name for the keystore that you created as described in "Generating Private Keys and Creating the Java Keystore" on page 10-9. This field defaults to `./default-keystore.jks`, which represents the default Java keystore name, `default-keystore.jks`, located in the `domain_name/config/fmwconfig` directory. If you used a different name or location for the keystore, enter that value instead.

   - In the **Password** and **Confirm Password** fields, enter the password for the keystore. This password must match the password you used when you created the keystore using the keytool utility, as described in "Generating Private Keys and Creating the Java Keystore" on page 10-9, for example *`password`*.

6. In the Identity Certificates section of the page, enter the alias and passwords for the signature and encryption keys as follows:

   - For the Signature Key, enter the alias name in the **Key Alias** field, and the password for the alias in the **Signature Password** and **Confirm Password** fields. The values you specify here must match the values in the keystore. For example, `orakey` and *`password`*.

   - For the Encryption Key, enter the alias name in the **Crypt Alias** field, and the password for the alias in the **Crypt Password** and **Confirm Password** fields.

The values you specify here must match the values in the keystore. For example, `orakey` and *`password`*.

The alias and password for the signature and encryption keys define the string alias and password used to store and retrieve the keys. These values are created in the credential store as `sign-csf-key` and `enc-csf-key`.

7. Click **OK** to submit the changes.

   If you used a file-based keystore provider, the changes require a server restart to take effect. A restart is not required for database or LDAP-based keystore service providers.

10.2.2.2 Using WLST

Follow these steps to configure the credential store to access the Oracle WSM keystore using WLST commands.

1. Go to the Oracle Common home directory for your installation, for example `/home/Oracle/Middleware/oracle_common`.

   For information about the Oracle Common home directory and installing Oracle Fusion Middleware, see the *Oracle Fusion Middleware Installation Planning Guide*.

2. Start WLST using the `WLST.sh/cmd` command located in the `oracle_common/common/bin` directory. For example:

   - `/home/Oracle/Middleware/oracle_common/common/bin/wlst.sh` (UNIX)
   - `C:\Oracle\Middleware\oracle_common\common\bin\wlst.cmd` (Windows)

   When executed, these commands start WLST in offline mode. To use the credential store WLST commands, you must use WLST in online mode.

3. Start Oracle WebLogic Server.

   For more information, see "Start and stop servers" in the *Oracle WebLogic Server Administration Console Help*.

4. Connect to the running WebLogic Server instance using the `connect()` command. For example, the following command connects WLST to the Administration Server at the URL `myAdminServer.example.com:7001` using the username/password credentials `weblogic`/*`password`*:

```
connect("weblogic","password","t3://myAdminServer.example.com:7001")
```

5. Enter the `createCred` command to create an entry in the credential store for the keystore name and password as follows:

```
createCred(map="oracle.wsm.security", key="keystore-csf-key", user="Oracle
WSM", password="password", desc="Keystore key")
```

   Note that you can enter any value for `user`. This field is ignored for the `keystore-csf-key` entry. The value of `password` must match the password that you specified when you created the keystore as described in "Generating Private Keys and Creating the Java Keystore" on page 10-9 (in this example *`password`*).

6. Enter the `createCred` command to create an entry in the credential store for the signature key alias and password as follows:

```
createCred(map="oracle.wsm.security", key="sign-csf-key", user="orakey",
password="password", desc="Signing key")
```

The values of `user` and `password` must match the alias name and password for the signature key in the keystore that you specified when you created the keystore as described in "Generating Private Keys and Creating the Java Keystore" on page 10-9. In this example, the values are `orakey` and *`password`*.).

7. Enter the `createCred` command to create an entry in the credential store for the encryption key alias and password as follows:

```
createCred(map="oracle.wsm.security", key="enc-csf-key", user="orakey",
password="password", desc="Encryption key")
```

The values of `user` and `password` must match the alias name and password for the encryption key in the keystore that you specified when you created the keystore as described in "Generating Private Keys and Creating the Java Keystore" on page 10-9. In this example, the values are `orakey` and `password`.).

10.2.3 Obtaining a Trusted Certificate and Importing it into the Keystore

You can obtain a certificate from a Certificate Authority (CA), such as Verisign or Entrust.net, and include it in the keystore. To get the certificate, you must create a Certificate Request and submit it to the CA. The CA will authenticate the certificate requestor and create a digital certificate based on the request.

To obtain a trusted certificate and import the certificate into the keystore:

1. Generate the private key and self-signed certificate. The self-signed certificate will be replaced by the trusted certificate.

> **Note:** If your keystore already contains a self-signed certificate that you created previously, as described in "Generating Private Keys and Creating the Java Keystore" on page 10-9, you can ignore this step and proceed to step 2.

Use the `keytool -genkeypair` command to generate the key pair for a specified alias, in this example `orakey`. It will create the keystore if it did not exist.

```
keytool -genkeypair -keyalg RSA -alias orakey -keypass password
-keystore default-keystore.jks -storepass password -validity 3600
```

2. Generate the certificate request.

Use the `keytool -certreq` command to generate the request. The following command generates a certificate request for the `orakey` alias and a Certificate Signing Request (CSR) named `certreq_file`.

```
keytool -certreq -alias orakey -sigalg "SHA1withRSA" -file certreq_file
-storetype jks -keystore default-keystore.jks
```

3. Submit the CSR file to a CA such as VeriSign, for example. The CA will authenticate the request and return a certificate or a certificate chain.

4. Import the CA root certificate which authenticates the CA's public key.

Use the `keytool -importcert` command to import the trusted CA root certificate (named `VerisignCAcert.cer` in this example), using the alias `verisignca` into the `default-keystore.jks` keystore. The keytool utility prompts for the needed password.

```
keytool -importcert -alias verisignca -trustcacerts -file
VerisignCAcert.cer -keystore default-keystore.jks
```

5. Replace the self-signed certificate with the trusted CA certificate issued by the CA in response to the certificate request.

   Use the `keytool -importcert` command. The following command replaces the self-signed certificate for the alias `orakey` with the trusted CA certificate named, in this example, `MyCertIssuedByVerisign.cer`. The keytool utility prompts for the needed password.

   ```
   keytool -importcert -trustcacerts -alias orakey -file
   MyCertIssuedByVerisign.cer -keystore default-keystore.jks
   ```

10.2.4 Setting Up the Web Service Client Keystore

You need to create a Java Key Store (JKS) keystore to store the signature and encryption keys required by the X.509 token on the client. Keys are used for a variety of purposes, including authentication and data integrity. For example:

- To sign data, you must have the signer's private key.
- To verify a signature, you must have a trusted CA certificate and the public key that matches the private key.
- To encrypt data, you must have the recipient's public key. The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.
- To decrypt data, you must have the private key that corresponds to the public key.

These trusted certificates and public and private keys are stored in the keystore. The following sections describe the requirements for the different types of message protection policies, how to create and use these keystores, and how to obtain trusted certificates.

- "How Different Security Policies Use Private Keys and Certificates" on page 10-3
- "Generating Private Keys and Creating the Java Keystore" on page 10-9
- "Obtaining a Trusted Certificate and Importing it into the Keystore" on page 10-15

10.2.5 Managing Java Keystore Certificates

You can use WebLogic Scripting Tool (WLST) commands to manage Java keystore (JKS) certificates.

- Listing all aliases within a keystore
- Displaying a certificate within a keystore
- Exporting and importing trusted certificates or a certificate chain associated with a private key:
  - Certificates - Base64 encoded X.509 (`.cer`)
  - Certificate Chain - PKCS7 (`.p7b`) or as an array of certificates
- Deleting certificates from the keystore

10.2.5.1 Using WLST

Follow these steps to manage JKS certificates using WLST commands.

1. Go to the Oracle Common home directory for your installation, for example `/home/Oracle/Middleware/oracle_common`.

For information about the Oracle Common home directory and installing Oracle Fusion Middleware, see the *Oracle Fusion Middleware Installation Planning Guide*.

2. Start WLST using the `WLST.sh/cmd` command located in the `oracle_common/common/bin` directory. For example:

   - `/home/Oracle/Middleware/oracle_common/common/bin/wlst.sh` (UNIX)
   - `C:\Oracle\Middleware\oracle_common\common\bin\wlst.cmd` (Windows)

   When executed, these commands start WLST in offline mode. To use the credential store WLST commands, you must use WLST in online mode.

3. Start Oracle WebLogic Server.

   For more information, see "Start and stop servers" in the *Oracle WebLogic Server Administration Console Help*.

4. Connect to the running WebLogic Server instance using the `connect()` command. For example, the following command connects WLST to the Administration Server at the URL `myAdminServer.example.com:7001` using the username/password credentials `weblogic/`password:

   ```
   connect("weblogic","password","t3://myAdminServer.example.com:7001")
   ```

5. Use these commands to manage your keystore:

   - **`displayWSMCertificate`** – displays the string representing the contents of a user's certificate if the alias specifies a `KeyStore.TrustedCertificateEntry`. Displays the certificates in the chain if the alias points to a certificate chain specified by the `KeyStore.PrivateKeyEntry`. For example:

     ```
     wls:/DefaultDomain/serverConfig> displayWSMCertificate('testalias')
     ```

   - **`listWSMKeystoreAliases`** – Lists all the aliases in the keystore. For example:

     ```
     wls:/DefaultDomain/serverConfig> listWSMKeystoreAliases()
     ```

   - **`deleteWSMKeyStoreEntry`** – Deletes a single `KeyStore.TrustedCertificateEntry` entry from the keystore. For example:

     ```
     wls:/DefaultDomain/serverConfig> deleteWSMKeyStoreEntry(alias='testalias')
     ```

   - **`deleteWSMKeyStoreEntries`** – Deletes all `KeyStore.TrustedCertificateEntry` entries from the keystore, except those identified by the aliases in the exclusion list.

     In this example, all key store entries are deleted from the keystore, except for the `testalias` and `testalias2` aliases, which are specified on the exclusion list:

     ```
     wls:/DefaultDomain/serverConfig>
     deleteWSMKeyStoreEntries(exclusionList=['testalias', 'testalias2'])
     ```

   - **`exportWSMKeyStoreCertificate`** – Exports a trusted certificate or a certificate chain associated with a private key, indicated by a specified alias, to a specified location.

     In this example, the trusted certificate `testalias` is identified by type as `Certificate` and is exported to the specified `certificate.cer` file:

     ```
     wls:/DefaultDomain/serverConfig>
     exportWSMCertificate(alias='testalias',certFile='/tmp/certificate.cer',type
     ='Certificate')
     ```

In this example, the certificate chain `testalias2` is identified by type as `PKCS7` and is exported to the specified `certificatechain.p7b` file:

```
wls:/base_domain/serverConfig>
exportWSMCertificate(alias='testalias2',certFile='/tmp/certificatechain.p7b
', type='PKCS7')
```

- **`importWSMKeyStoreCertificate`** – Imports a trusted certificate or a certificate chain associated with a private key indicated by the specified alias. The Base64 encoded certificate will be imported from the specified location.

  In this example, the trusted certificate `testalias` is identified by type as `Certificate` and is imported from the specified `certificate.cer` file:

  ```
  wls:/DefaultDomain/serverConfig>
  importWSMCertificate(alias='testalias',certFile='/tmp/certificate.cer',type
  ='Certificate')
  ```

  In this example, the certificate chain `testalias` is identified by type as `PKCS7` and is imported from the specified `certificatechain.p7b` file:

  ```
  wls:/base_domain/serverConfig>
  importWSMCertificate(alias='testalias',certFile='/tmp/certificatechain.p7b'
  ,type='PKCS7')
  ```

For more information about these WLST commands, see "Web Services Custom WLST Commands" in the *WebLogic Scripting Tool Command Reference*.

10.3 Configuring the Credential Store

> **Note:** To configure the credential store using the REST API, see "Credential Store Management" in *REST API for Managing Credentials and Keystores with Oracle Web Services Manager*.

Oracle WSM uses the Credential Store Framework (CSF) to manage the credentials in a secure form. The CSF provides a way to store, retrieve, and delete credentials for a Web Service and other applications. Oracle WSM uses the credential store to look up the following:

- Alias names and passwords for keys in the Java keystore

  For details about how Oracle WSM uses the credential store to look up alias names and passwords from the Java keystore, see "How Oracle WSM Locates Keystore And Key Passwords" on page 10-23.

- Usernames and passwords used for authentication

  Suppose, for example, that you have a Web service that accepts a username token for authentication. If you create a Web service client to talk to this Web service, you need to configure the Web service client with a username and password that can be sent to the Web service. You store this username and password in the credential store (using either Fusion Middleware Control or WLST) and assign it a csf key.

  For example, the `oracle/wss_username_token_client_policy` policy includes the `csf-key` property, with a default value of `basic.credentials`. To use the wss_username_token_client_policy, you should create a new password credential in the CSF using the credential name `basic.credentials`, and the username and password with which the client needs to connect. If you have two Web service clients that use this same client policy, these clients can either share the same password credential, which defaults to basic.credentials, or each one can have its own credential. In the latter case, you need to create two password credentials in the CSF, for example App1.credentials and App2.credentials, for Client1 and Client2 respectively. For Client1, you set the csf-key configuration override to App1.credentials, and for Client2, you set the csf-key property to App2.credentials. For more information, see "Attaching Client Policies Permitting Overrides" on page 8-31. Note that in both cases, the usernames and passwords must represent valid users in the OPSS identity store.

A password credential can store a username and password. A generic credential can store any credential object.

The CSF configuration is maintained in the jps-config.xml file in the *domain-home*/config/fmwconfig directory.

When you configure the Oracle WSM keystore using Fusion Middleware Control, as described in "Configuring the Oracle WSM Keystore" on page 10-11, the aliases and passwords that you specify are securely stored in the credential store. If, however, you add other aliases to the keystore, or you need to add authentication credentials for a client, you need to ensure that they are configured and stored in the credential store also, as described in the following section.

10.3.1 Adding Keys and User Credentials to the Credential Store

You can use Fusion Middleware Control or WLST commands to add keys and user credentials to the credential store. Both methods are described in the following procedures.

Note: The example procedures in this section describe how to add user credentials for the basic.credentials key as described above, and the example ServiceA and ServiceB aliases described in "Advanced Setup Considerations" on page 10-8. In your own environment, you should use aliases and passwords that are appropriate for your configuration.

Before adding key credentials to the credential store, ensure that the private keys and aliases exist in the keystore. You can create them using commands such as the following:

```
keytool -genkeypair -keyalg RSA -alias ServiceA -keypass password
-keystore default-keystore.jks -storepass password -validity 3600

keytool -genkeypair -keyalg RSA -alias ServiceB -keypass welcome3
-keystore default-keystore.jks -storepass password -validity 3600
```

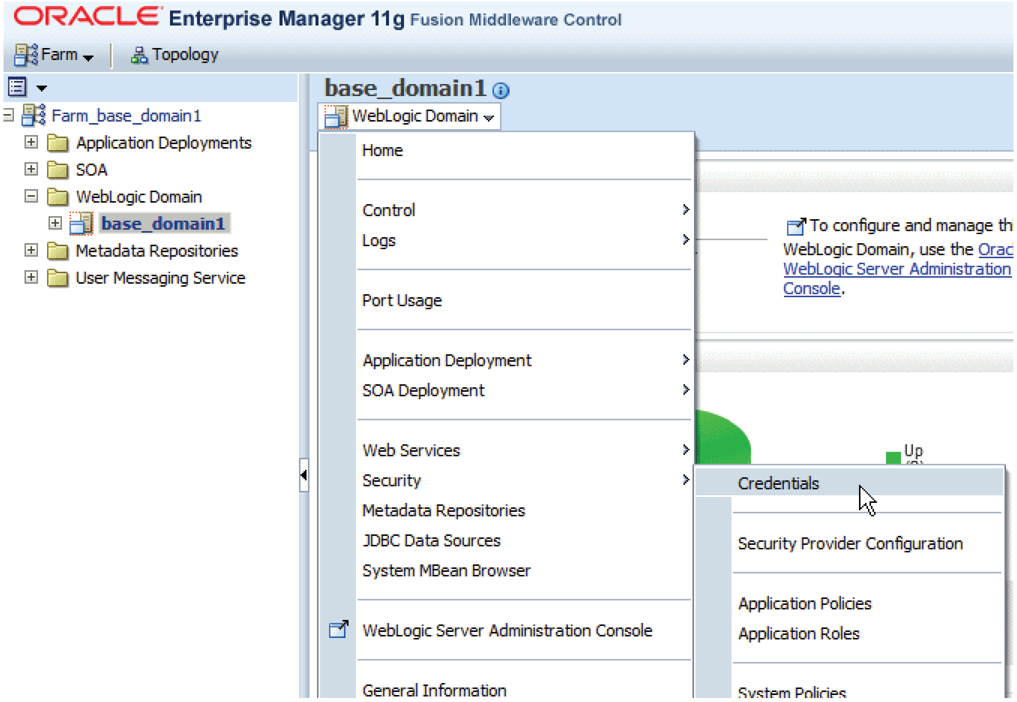
For more information about the keystore, see "Generating Private Keys and Creating the Java Keystore" on page 10-9.

10.3.1.1 Using Fusion Middleware Control

Follow these steps in Fusion Middleware Control to add keys and certificates to the credential store:

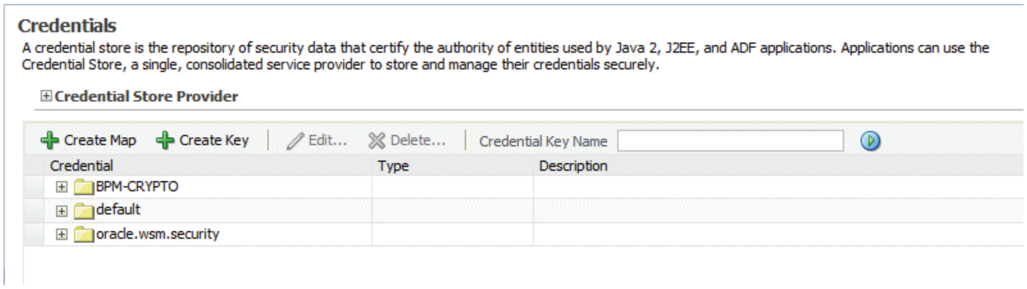
1. In the Navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the keystore. Select the domain.
2. From the **WebLogic Domain** menu, select **Security** then **Credentials**.

Figure 10–4 Credential Store Menu



The Credentials page is displayed, as shown in Figure 10–5.

Figure 10–5 Credential Store Provider Configuration Page


Credentials

A credential store is the repository of security data that certify the authority of entities used by Java 2, J2EE, and ADF applications. Applications can use the Credential Store, a single, consolidated service provider to store and manage their credentials securely.

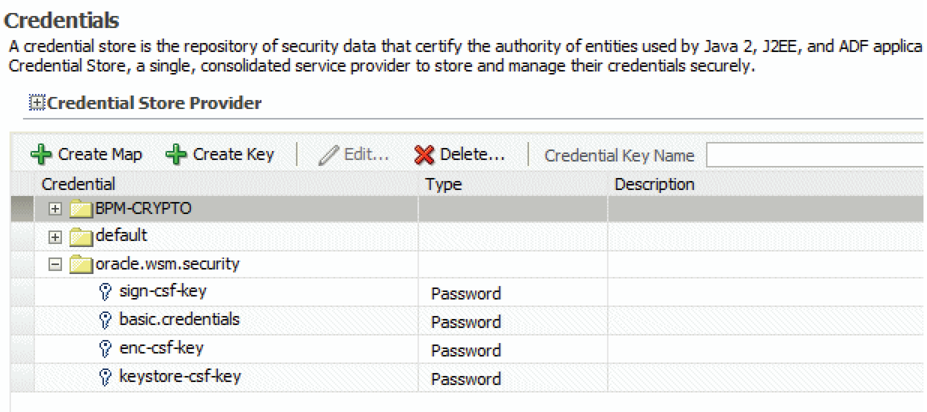
⊟ Credential Store Provider

Credential	Type	Description
⊞ BPM-CRYPTO		
⊞ default		
⊞ oracle.wsm.security		

Note that in this configuration, the `oracle.wsm.security` credential map already exists in the credential store. This credential map was created when you configured the Oracle WSM keystore as described in "Configuring the Oracle WSM Keystore" on page 10-11. If you do not see this credential map in your configuration, you can create it by clicking the **Create Map** button, and entering `oracle.wsm.security` in the **Map Name** field.

3. Optionally, expand the `oracle.wsm.security` map in the Credential table to view the keys that have been configured in the map. Figure 10–6 illustrates a sample Oracle WSM credential store configuration.

Figure 10–6 Keys Configured in Oracle WSM Credential Map



Credentials

A credential store is the repository of security data that certify the authority of entities used by Java 2, J2EE, and ADF applica
Credential Store, a single, consolidated service provider to store and manage their credentials securely.

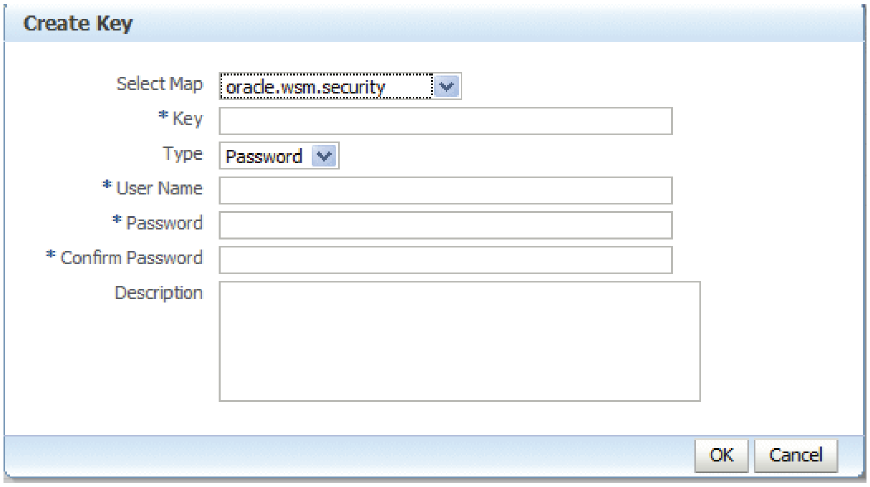
Credential Store Provider

Credential	Type	Description
⊞ BPM-CRYPTO		
⊞ default		
⊟ oracle.wsm.security		
sign-csf-key	Password	
basic.credentials	Password	
enc-csf-key	Password	
keystore-csf-key	Password	

You can edit the keys in the credential map by selecting the key and clicking **Edit**. Make sure that any changes you make in the credential store are consistent with the definition of the key in the Oracle WSM Java keystore.

4. Click **Create Key** to create new entries in the `oracle.wsm.security` credential map, for example for the `ServiceA` and `ServiceB` aliases. The Create Key dialog box appears, as shown in Figure 10–7.

Figure 10–7 Create Key Dialog Box



Create Key

Select Map	oracle.wsm.security
* Key	
Type	Password
* User Name	
* Password	
* Confirm Password	
Description	

a. From the **Select Map** menu, select the map name **oracle.wsm.security** if it is not already selected.

b. In the **Key** field, enter `csfServiceA` to create a key-value pair to access the key store.

c. From the **Type** menu, select **Password**.

d. In the **User Name** field, enter the alias name that you specified for the private key in the keystore, for example `ServiceA`.

e. In the **Password** and **Confirm Password** fields, enter the password that you specified for the alias in the keystore, for example *`password`*.

f. In the **Description** field, enter a description of for the entry, for example, `Key for ServiceA`.

g. Click **OK**.

h. Click **Create Key** again and provide the values for any additional keystore aliases, such as `csfServiceB` for the `ServiceB` alias.

5. Optionally, click **Create Key** to create entries in the `oracle.wsm.security` credential map for the any `csf-key` user credentials, for example `basic.credentials`, as follows:

   a. From the **Select Map** menu, select the map name **oracle.wsm.security** if it is not already selected.

   b. In the **Key** field, enter `basic.credentials`. In this example, we use `basic.credentials` but you can specify any name you choose for the key.

   c. From the **Type** menu, select **Password**.

   d. In the **User Name** field, enter a valid username that exists in the OPSS identity store, for example `AppID`.

   e. In the **Password** and **Confirm Password** fields, enter a valid password for the user, for example *password*.

   f. In the **Description** field, enter a description of for the entry, for example, `Username and Password for basic.credential key`.

   g. Click **OK**.

6. Restart the server.

10.3.1.2 Using WLST

Follow these steps to add additional keys and user credentials to the credential store using WLST commands.

1. Go to the Oracle Common home directory for your installation, for example `/home/Oracle/Middleware/oracle_common`.

   For information about the Oracle Common home directory and installing Oracle Fusion Middleware, see the *Oracle Fusion Middleware Installation Planning Guide*.

2. Start WLST using the `WLST.sh/cmd` command located in the `oracle_common/common/bin` directory. For example:

   - `/home/Oracle/Middleware/oracle_common/common/bin/wlst.sh` (UNIX)
   - `C:\Oracle\Middleware\oracle_common\common\bin\wlst.cmd` (Windows)

   When executed, these commands start WLST in offline mode. To use the credential store WLST commands, you must use WLST in online mode.

3. Start Oracle WebLogic Server.

   For more information, see "Start and stop servers" in the *Oracle WebLogic Server Administration Console Help*.

4. Connect to the running WebLogic Server instance using the `connect()` command. For example, the following command connects WLST to the Administration Server at the URL `myAdminServer.example.com:7001` using the username/password credentials `weblogic/`*`password`*:

```
connect("weblogic","password","t3://myAdminServer.example.com:7001")
```

5. Use the `createCred` command to create entries in the `oracle.wsm.security` credential map for the `ServiceA` and `ServiceB` aliases. For example, create an entry `csfServiceA` for the `ServiceA` alias, using a command such as the following:

```
wls:/DefaultDomain/serverConfig> createCred(map="oracle.wsm.security",
key="csfServiceA", user="ServiceA", password="password", desc="Key for
ServiceA")
```

6. Repeat step 5 to create an entry for any additional aliases, for example `csfServiceB`, for the `ServiceB` alias.
7. Use the `createCred` command to create entries in the `oracle.wsm.security` credential map for the any `csf-key` user credentials, for example `basic.credentials`.

```
wls:/DefaultDomain/serverConfig> createCred(map="oracle.wsm.security",
key="basic.credentials", user="AppID", password="password", desc="Key for
ServiceA")
```

10.3.2 How Oracle WSM Locates Keystore And Key Passwords

Oracle WSM expects keystore and key passwords to be in the Credential Store Framework (CSF). Here is how it works.

- A JKS keystore file is protected by a keystore password.
- A keystore file consists of zero or more private keys, and zero or more trusted certificates. Each private key has its own password, (although it is common to set the key passwords to be the same as the keystore password). Oracle WSM needs to know both the keystore password and key password.
- The CSF consists of many *maps*, each with a distinct name. Oracle WSM only uses the map `oracle.wsm.security`.
- Inside each map is a mapping from multiple csf-key entries to corresponding credentials. A csf-key is just a simple name, but there can be many different types of credentials. The most common type of credential is a password credential which is primarily comprised of a username and a password.

  Oracle WSM refers to the following csf-keys inside the `oracle.wsm.security` map:

  – `keystore-csf-key` - This key should contain the keystore password. The username is ignored.

  – `enc-csf-key` - This key should contain the encryption key alias as the username, and the corresponding key password.

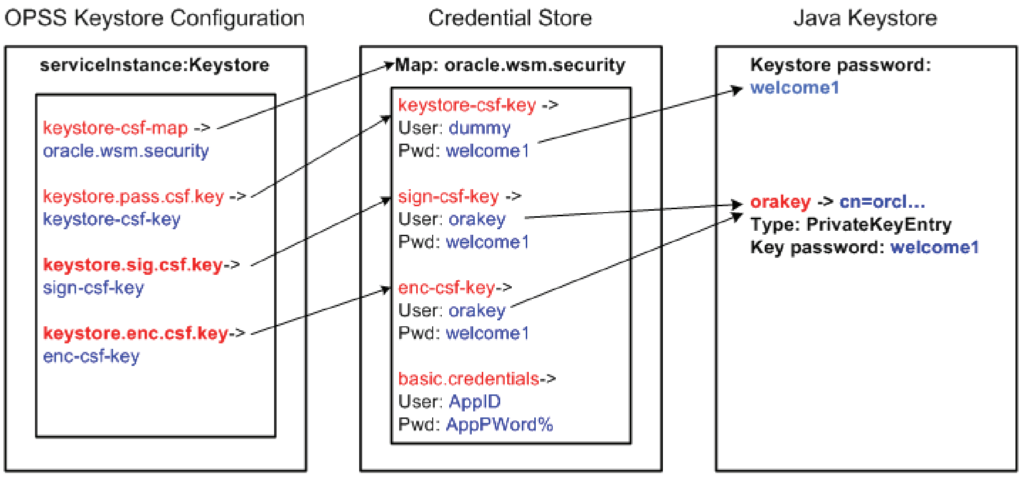
  – `sign-csf-key` - This key should contain the signature key alias as the username, and the corresponding key password.

  In addition to these csf-keys, you should add a csf-key entry for every new private key that you want Oracle WSM to use, for example when you want to specify signature and encryption keys in configuration overrides.

Figure 10–8 illustrates the relationship between the keystore configuration in the OPSS, the `oracle.wsm.security` map in the credential store, and the Oracle WSM Java keystore.

Note: For federated environments, an application-level credential map name can be set by an administrator as a csf.map configuration override on certain predefined policies and assertion templates. For more information, see "Creating an Application-level Credential Map" on page 10-24.

Figure 10–8   Oracle WSM Java Keystore Configuration for Message Protection


OPSS Keystore Configuration
Credential Store
Java Keystore
serviceInstance:Keystore
keystore-csf-map -> oracle.wsm.security
keystore.pass.csf.key -> keystore-csf-key
keystore.sig.csf.key-> sign-csf-key
keystore.enc.csf.key-> enc-csf-key
Map: oracle.wsm.security
keystore-csf-key -> User: dummy Pwd: welcome1
sign-csf-key -> User: orakey Pwd: welcome1
enc-csf-key-> User: orakey Pwd: welcome1
basic.credentials-> User: AppID Pwd: AppPWord%
Keystore password: welcome1
orakey -> cn=orcl...
Type: PrivateKeyEntry
Key password: welcome1

As shown in the figure:

- The `keystore.csf.map` property points to the Oracle WSM map in the credential store that contains the CSF aliases. In this case `keystore.csf.map` is defined as the recommended name `oracle.wsm.security`, but it can be any value.
- The `keystore.pass.csf.key` property points to the CSF alias `keystore-csf-key` that is mapped to the username and password of the keystore. Only the password is used; username is redundant in the case of the keystore.
- The `keystore.sig.csf.key` property points to the CSF alias `sign-csf-key` that is mapped to the username and password of the private key that is used for signing.
- The `keystore.enc.csf.key` property points to the CSF alias `enc-csf-key` that is mapped to the username and password of the private key that is used for decryption.

10.4 Creating an Application-level Credential Map

An application-level credential map name can be set in certain predefined policies using the `csf.map` configuration property, which can be used to override the domain-level credential map on a per-attachment basis. The `csf.map` configuration override is available in all policies and assertion templates that have either a csf-key or keystore-related csf keys.

For more information about configuring overrides, see "Attaching Client Policies Permitting Overrides" on page 8-31.

For a list of the available predefined policies, see Policies that Can Be Used to Access an Application-level CSF Map.

10.4.1 How CSF Keys Are Retrieved from an Application-level Credential Map

The domain-level `oracle.wsm.security` credential map is created in the credential store when you configure the Oracle WSM keystore, as described in "Configuring the Oracle WSM Keystore" on page 10-11.

When an application-level credential map is configured, then the client csf-keys (`csf-key` and user keys `sts.auth.user.csf.key`) are retrieved only from that map, and not the domain-level map, as follows:

- If an application-level csf map is configured, csf keys are retrieved from it. An exception is thrown if the csf key is not found.
- If an application-level map is not configured, csf keys are retrieved from the domain-level csf map. An exception is thrown if the csf key is not found.

Note that the behavior is different for shared cfs-keys, as follows:

- `keystore-csf-key` – This csf key will always be retrieved from the domain-level credential map (`oracle.wsm.security`).
- `enc-csf-key` – If an application-level map is configured, this csf key will be retrieved from it first; if not found, then from the domain-level csf map.
- `sign-csf-key` – If an application-level map is configured, this csf key will be retrieved from it first; if not found, then from the domain-level csf map.

10.4.2 Granting Permission to Access an Application-level Credential Map

In order to access an application-level credential map, you must:

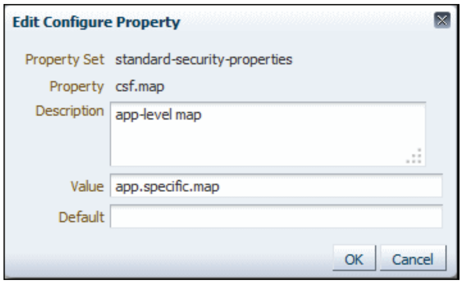
- Configure the csf.map Property Override
- Grant CredentialAccessPermission to wsm-agent-core.jar
- Grant WSIdentityPermission to wsm-agent-core.jar

10.4.2.1 Configure the csf.map Property Override

For an application to access its own credential map, the `csf.map` configuration override must be set for the policy that is attached to the application.

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the policy for which you want to edit a property from the Policies table and click **Edit**.
3. On the Edit Policy page, click the **Configurations** tab.
4. Select the **csf.map** configuration property and click **Edit**. The Edit Configure Property dialog box shown in Figure 10–9 appears.

Figure 10–9 Edit Configure Property Dialog



Edit Configure Property

Property Set: standard-security-properties

Property: csf.map

Description: app-level map

Value: app.specific.map

Default:

5. In the value field, enter the name of the application-level map for the policy to use and click **OK**.
6. Validate the policy.
7. Click **Save**.

10.4.2.2 Grant CredentialAccessPermission to wsm-agent-core.jar

Grant `CredentialAccessPermission` to the `wsm-agent-core.jar`. This permission is required for Oracle Platform Security Services (OPSS) to allow access to the credential map in the CSF store.

Using Oracle Enterprise Manager

You can grant `CredentialAccessPermission` to an application-level credential map from the domain's System Policies page.

1. In the Navigator pane, expand **WebLogic Domain** to select the domain that you want to configure a new system policy in.
2. From the **WebLogic Domain** menu, select **Security** then **System Policies**.
3. Click **Create** to open the Create System Grant page.
4. If necessary, in the Grant To drop box, select **Codebase** as the policy type.
5. Enter the following string in the Codebase field:

   ```
   file:${common.components.home}/modules/oracle.wsm.agent.common_
   ${jrf.version}/wsm-agent-core.jar
   ```

6. Click **Add** above the Permissions table.
7. In the Add Permission dialog, click the **Select here to enter details for a new permission** check box, and enter the following information:

   Permission Class –
   `oracle.security.jps.service.credstore.CredentialAccessPermission`

   Resource Name –
   `context=SYSTEM,mapName=application.csf.map.name,keyName=*`

   Where `mapName` is the name of the application-level credential map that needs to be configured.

   Permission Actions – `*`

8. Click **OK** to return to the Create System Grant page. The selected permission is added to the table Permissions.
9. Click **OK** to return to the System Policies page. A message at the top of the page informs you the result of the operation. If successful, the policy is added to the table at the bottom of the page.

For more information about configuring system policies, see "Managing System Policies" in *Oracle Fusion Middleware Application Security Guide*.

Using WLST

You can grant `CredentialAccessPermission` to an application-level credential map using the `grantPermission` WLST command.

1. Start WLST and connect to the running WebLogic Server instance, as described in Using WLST.
2. Use the `grantPermission` command to create the codebase system policy:

   ```
   grantPermission(appStripe=None,
   codeBaseURL='file:${common.components.home}/modules/oracle.wsm.agent.common_
   ${jrf.version}/wsm-agent-core.jar',principalClass=None,principalName=None,permC
   lass='oracle.security.jps.service.credstore.CredentialAccessPermission',permTar
   get='context=SYSTEM,mapName=application.csf.map.name,keyName=*',permActions='*'
   ```

```
)
```

For more information about this WLST command, see "Infrastructure Security Custom WLST Commands" in the *WebLogic Scripting Tool Command Reference*.

10.4.2.3 Grant WSIdentityPermission to wsm-agent-core.jar

Grant `WSIdentityPermission` to `wsm-agent-core.jar` by using the `mapName` term and the `getKey` action. If this permission is not given to a particular application, it will not be able to access the application-level credential map.

Using Oracle Enterprise Manager

You can grant `CredentialAccessPermission` to an application-level credential map from the domain's System Policies page.

1. In the Navigator pane, expand **WebLogic Domain** to select the domain that you want to configure a new system policy in.
2. From the **WebLogic Domain** menu, select **Security** then **System Policies**.
3. If you already completed the steps in Grant CredentialAccessPermission to wsm-agent-core.jar, in the Search section, select **Codebase** as the type and search for the following string in the Name field:

   ```
   file:${common.components.home}/modules/oracle.wsm.agent.common_
   ${jrf.version}/wsm-agent-core.jar
   ```

4. Select the codebase grant in the Search table and click **Edit**.
5. On the Edit System Grant page, click **Add** above the Permissions table.
6. On the Add Permission dialog, click the **Select here to enter details for a new permission** check box, and enter the following information:

   Permission Class – `oracle.wsm.security.WSIdentityPermission`

   Resource Name – `resource=usermessagingserver,mapName=application.specific.map`

   Where `resource` is the name of the application for which permission is required and `mapName` is the name of the application-level credential map that needs to be configured.

   Permission Actions – `getKey`

7. Click **OK** to return to the Create System Grant page. The selected permission is added to the table Permissions.
8. Click **OK** to return to the System Policies page. A message at the top of the page informs you the result of the operation and the new permissions added for codebase grant.

For more information about configuring system policies, see "Managing System Policies" in *Oracle Fusion Middleware Application Security Guide*.

Using WLST

You can grant `CredentialAccessPermission` to an application-level credential map using the `grantPermission` WLST command.

1. Start WLST and connect to the running WebLogic Server instance, as described in Using WLST.
2. Use the `grantPermission` command to create the codebase system policy:

```
grantPermission(appStripe=None,
codeBaseURL='file:${common.components.home}/modules/oracle.wsm.agent.common_
${jrf.version}/wsm-agent-core.jar',principalClass=None,principalName=None,permC
lass='oracle.wsm.security.WSIdentityPermission',permTarget='resource=usermessag
ingserver,mapName=application.specific.map',permActions='getKey')
```

For more information about this WLST command, see "Infrastructure Security Custom WLST Commands" in the *WebLogic Scripting Tool Command Reference*.

Example of Granting Permission for Application-level In system-jazn-data.xml

Here is an example of granting permission in `system-jazn-data.xml` when used to access an application-level credential map and restrict access to a specific application. `resource` is the name of application for which application credential map needs to be configured and `mapName` is the name of the application-level credential map that needs to be configured.

```
<grant>
   <grantee>
      <codesource>
         <url>
file:${common.components.home}/modules/oracle.wsm.agent.common_
${jrf.version}/wsm-agent-core.jar
         </url>
      </codesource>
   </grantee>
   <permissions>
      <permission>
            <class>oracle.wsm.security.WSIdentityPermission</class>

<name>resource=usermessagingserver,mapName=application.specific.map</name>
            <actions>getKey</actions>
      </permission>
    </permissions>
</grant>
```

The `resource` term and `mapName` term also support asterisk (*) wildcards. Here are some examples of legal permission names when the action is `getKey`:

- `resource=usermessagingserver,mapName=application.specific.map`

  Only the `usermessagingserver` application can access the credential map `application.specific.map`.

- `resource=*,mapName=application.specific.map`

  All applications can access the credential map `application.specific.map`.

- `resource=usermessagingserver,mapName=*`

  The application `usermessagingserver` can access all credential maps.

- `resource=usermessagingserver,mapName=intel-*`

  The application `usermessagingserver` can access all credential maps that start with `intel-`.

- `resource=intel-*,mapName=application.specific.map`

  All applications that have a name starting with `intel-*` can access the credential map `application.specific.map`.

Note: When using `WSMIdentityPermission` to access an application-level credential map:

- The permissions are checked only for managed applications. For Java SE applications, permissions are not checked.
- The permissions do not work in Oracle Java Cloud Service environments where `java.security.AllPermission` is given to Oracle WSM JARs.
- The permission is not required for accessing the domain-level credential map. See "Adding Keys and User Credentials to the Credential Store" on page 10-19

10.4.3 Policies that Can Be Used to Access an Application-level CSF Map

An application-level credential map name can be set in the following predefined policies using the `csf.map` configuration override property, enabling you to override the domain-level credential map on a per-attachment basis.

For more information about predefined security assertion templates that contain the `csf.map` configuration property, see "Security Assertion Templates" on page C-1.

- http_basic_auth_over_ssl_client_policy
- http_jwt_token_client_policy
- http_jwt_token_identity_switch_client_policy
- http_jwt_token_over_ssl_client_policy
- http_jwt_token_over_ssl_service_policy
- http_jwt_token_service_policy
- http_oauth2_token_client_policy
- http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_policy
- http_oauth2_token_identity_switch_over_ssl_client_policy
- http_oauth2_token_opc_oauth2_client_policy
- http_oauth2_token_opc_oauth2_over_ssl_client_policy
- http_oauth2_token_over_ssl_client_policy
- oauth2_config_client_policy
- http_saml20_token_bearer_client_policy
- http_saml20_token_bearer_over_ssl_client_policy
- multi_token_over_ssl_rest_service_policy
- multi_token_rest_service_policy
- wss10_message_protection_client_policy
- wss10_message_protection_service_policy
- wss10_saml20_token_client_policy
- wss10_saml20_token_with_message_protection_client_policy
- wss10_saml20_token_with_message_protection_service_policy
- wss10_saml_hok_token_with_message_protection_client_policy

- wss10_saml_hok_token_with_message_protection_service_policy
- wss10_saml_token_client_policy
- wss10_saml_token_with_message_integrity_client_policy
- wss10_saml_token_with_message_integrity_service_policy
- wss10_saml_token_with_message_protection_client_policy
- wss10_saml_token_with_message_protection_service_policy
- wss10_saml_token_with_message_protection_ski_basic256_client_policy
- wss10_saml_token_with_message_protection_ski_basic256_service_policy
- wss10_username_id_propagation_with_msg_protection_client_policy
- wss10_username_id_propagation_with_msg_protection_service_policy
- wss10_username_token_with_message_protection_client_policy
- wss10_username_token_with_message_protection_service_policy
- wss10_username_token_with_message_protection_ski_basic256_client_policy
- wss10_username_token_with_message_protection_ski_basic256_service_policy
- wss10_x509_token_with_message_protection_client_policy
- wss10_x509_token_with_message_protection_service_policy
- wss11_message_protection_client_policy
- wss11_message_protection_service_policy
- wss11_saml20_token_with_message_protection_client_policy
- wss11_saml20_token_with_message_protection_service_policy
- wss11_saml_or_username_token_with_message_protection_service_policy
- wss11_saml_token_identity_switch_with_message_protection_client_policy
- wss11_saml_token_with_message_protection_client_policy
- wss11_saml_token_with_message_protection_service_policy
- wss11_sts_issued_saml_hok_with_message_protection_client_policy
- wss11_sts_issued_saml_hok_with_message_protection_service_policy
- wss11_sts_issued_saml_with_message_protection_client_policy
- wss11_username_token_with_message_protection_client_policy
- wss11_username_token_with_message_protection_service_policy
- wss11_x509_token_with_message_protection_client_policy
- wss11_x509_token_with_message_protection_service_policy
- wss_http_token_client_policy
- wss_http_token_over_ssl_client_policy
- wss_saml20_token_bearer_over_ssl_client_policy
- wss_saml20_token_over_ssl_client_policy
- wss_saml_token_bearer_client_policy
- wss_saml_token_bearer_over_ssl_client_policy

- wss_saml_token_bearer_identity_switch_client_policy
- wss_saml_token_over_ssl_client_policy
- wss_sts_issued_saml_bearer_token_over_ssl_client_policy
- wss_username_token_client_policy
- wss_username_token_over_ssl_client_policy

10.5 Configuring the OPSS Keystore Service for Message Protection

Note: To configure the OPSS keystore service for message protection using the REST API, see "KSS Keystore Management" in *REST API for Managing Credentials and Keystores with Oracle Web Services Manager*.

As described in "Managing Keys and Certificates with the Keystore Service" in *Oracle Fusion Middleware Application Security Guide*, the OPSS Keystore Service provides an alternate mechanism to manage keys and certificates for message security. You use the OPSS Keystore Service to create and maintain keystores of type *KSS*.

Note: Keystores can be exported and imported. Migration is supported for JKS and JCEKS certificate formats.

There is a single Oracle WSM keystore per domain, and it is shared by all Web services and clients running in the domain. Therefore, if you choose to configure the KSS keystore as described in this section, Oracle WSM uses only that KSS keystore and ignores any JKS keystores you might have also defined.

You can perform the OPSS Keystore Service operations using both Fusion Middleware Control and WLST. This section focuses on the Fusion Middleware Control steps, but "Managing Keys and Certificates with the Keystore Service" describes both options.

Perform these steps to configure an OPSS Keystore Service for message protection:

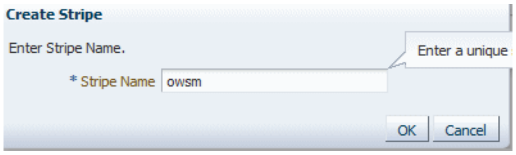
1. Create a stripe and name it `Oracle WSM`. (See "Creating a Keystore with Fusion Middleware Control" in *Oracle Fusion Middleware Application Security Guide* for more information.)

   a. From the **WebLogic Domain** menu, select **Security** then **Keystore**.

   b. Click **Create Stripe**. The Create Stripe screen is shown in Figure 10–10.

   c. Enter `Oracle WSM` and click **OK**. You must use this name.

Figure 10–10 Create Stripe



Create Stripe

Enter Stripe Name.

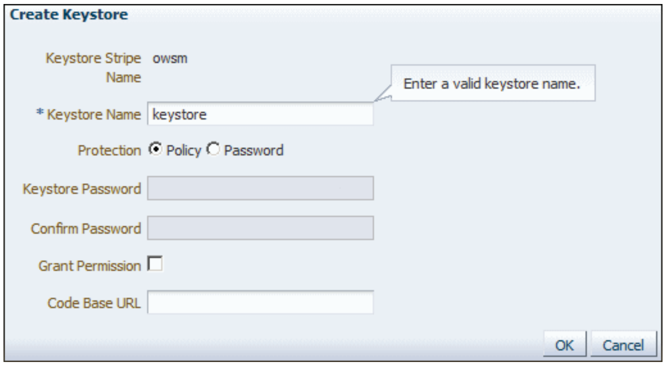
* Stripe Name: owsm

2. Create a keystore named `keystore` in the `Oracle WSM` stripe. (See "Creating a Keystore with Fusion Middleware Control" in *Oracle Fusion Middleware Application Security Guide* for more information.)

   a. Select the `Oracle WSM` stripe you created and click **Create Keystore**.

The Create Keystore page is shown in Figure 10–11.

Figure 10–11 Create Keystore



Create Keystore

Keystore Stripe Name owsm

Enter a valid keystore name.

* Keystore Name keystore

Protection ● Policy ○ Password

Keystore Password

Confirm Password

Grant Permission □

Code Base URL

b. Name this keystore `keystore`. You must use this name.

c. Set the protection type to Policy.

d. Clear the **Grant Permission** check box.

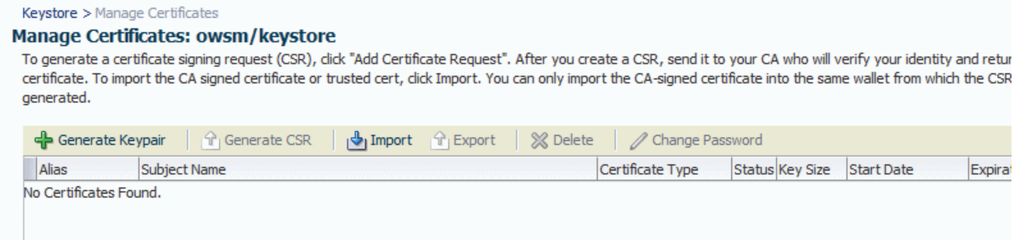
e. Do not specify a value in the **Code Base URL** field.

f. Click **OK**.

3. Select the keystore you just created and click **Manage**.

   The Manage Certificates screen is shown in Figure 10–12.

Figure 10–12 Manage Certificates



Keystore > Manage Certificates

Manage Certificates: owsm/keystore

To generate a certificate signing request (CSR), click "Add Certificate Request". After you create a CSR, send it to your CA who will verify your identity and retur certificate. To import the CA signed certificate or trusted cert, click Import. You can only import the CA-signed certificate into the same wallet from which the CSR generated.

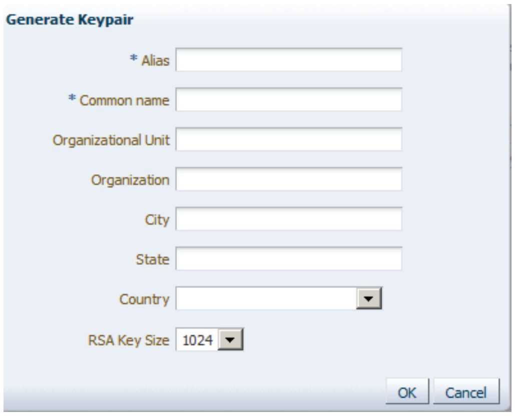
Alias	Subject Name	Certificate Type	Status	Key Size	Start Date	Expira
No Certificates Found.						

4. Click **Generate Keypair** to generate a private/public key pair.

   You typically use this keypair to both sign and encrypt requests. However, you can create separate key pairs for signing and encryption if you so choose.

   The Generate Keypair screen is shown in Figure 10–13.

Figure 10–13 Generate Keypair



RSA Key Size 1024

a. Specify an alias such as `orakey` for the key pair.

b. Specify other site-specific information as appropriate.

c. You can accept the default RSA key size if appropriate for your environment. Oracle requires a key length of 1024 bits or larger.

d. Click **OK**.

5. The certificate is generated by default as being issued by `CN=CertGenCAB, OU=FOR TESTING ONLY, O=MyOrganization, L=MyTown, ST=MyState, C=US`. This issuer does not exist in the keystore in the `Oracle WSM` stripe. Before you can use this certificate, you must do one of the following:

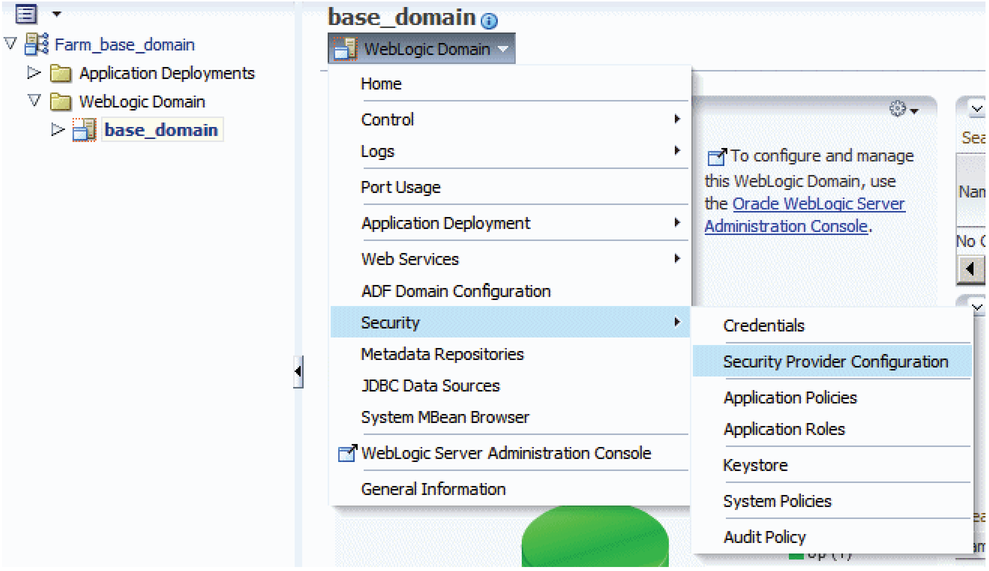
   - Export the `democa` certificate from the `castore` keystore in the `system` stripe and import it into the `keystore` keystore in the `Oracle WSM` stripe as a trusted certificate.

   - Export the certificate you generated to a file, sign it using a CA, and import the CA as well as the certificate into the `keystore` keystore in the `Oracle WSM` stripe.

6. Configure Oracle WSM to use this keystore and alias.

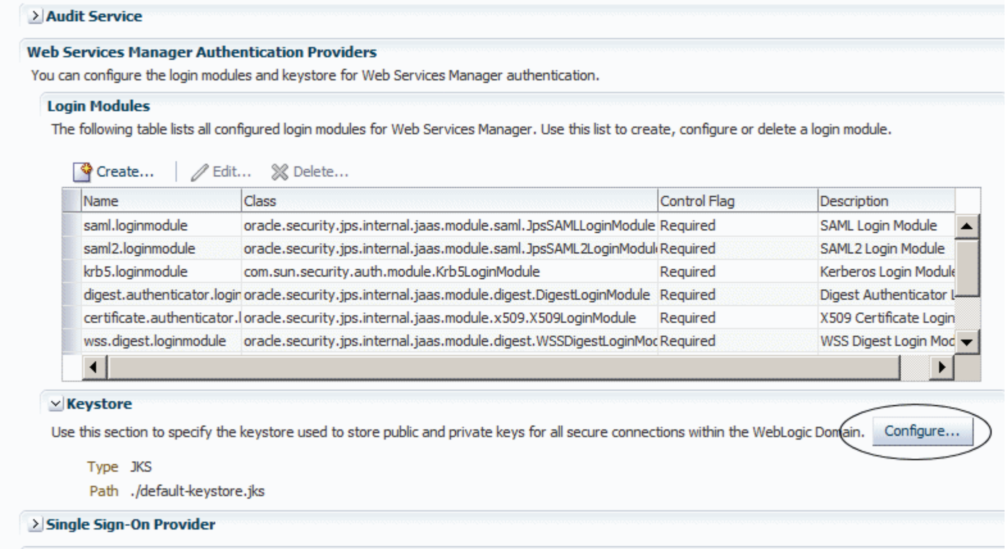
   From the **WebLogic Domain** menu, select **Security** then **Security Provider Configuration**, as shown in Figure 10–14.

Figure 10–14 Security Provider Configuration Menu



The Security Provider Configuration page is displayed, as shown in Figure 10–15.

Figure 10–15 Security Provider Configuration Page



Audit Service

Web Services Manager Authentication Providers

You can configure the login modules and keystore for Web Services Manager authentication.

Login Modules

The following table lists all configured login modules for Web Services Manager. Use this list to create, configure or delete a login module.

Name	Class	Control Flag	Description
saml.loginmodule	oracle.security.jps.internal.jaas.module.saml.JpsSAMLLoginModule	Required	SAML Login Module
saml2.loginmodule	oracle.security.jps.internal.jaas.module.saml.JpsSAML2LoginModul	Required	SAML2 Login Module
krb5.loginmodule	com.sun.security.auth.module.Krb5LoginModule	Required	Kerberos Login Module
digest.authenticator.logir	oracle.security.jps.internal.jaas.module.digest.DigestLoginModule	Required	Digest Authenticator L
certificate.authenticator.l	oracle.security.jps.internal.jaas.module.x509.X509LoginModule	Required	X509 Certificate Login
wss.digest.loginmodule	oracle.security.jps.internal.jaas.module.digest.WSSDigestLoginMoc	Required	WSS Digest Login Moc

Keystore

Use this section to specify the keystore used to store public and private keys for all secure connections within the WebLogic Domain. Configure...

Type JKS

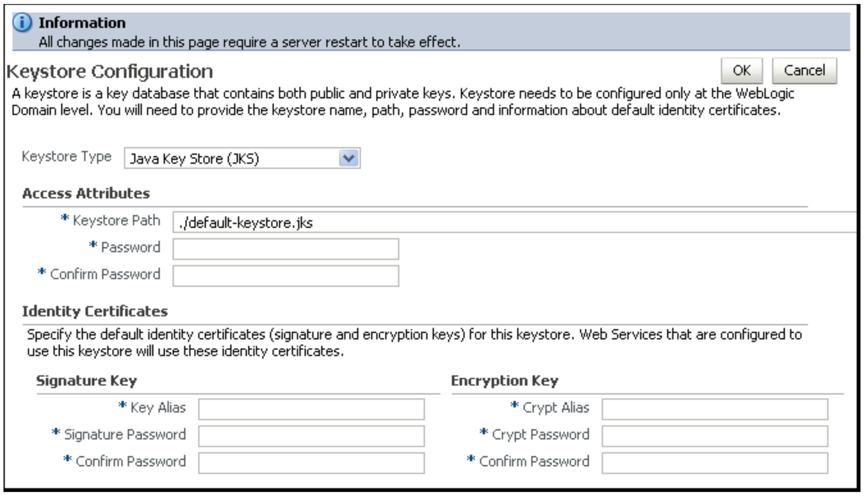
Path ./default-keystore.jks

Single Sign-On Provider

7. Click **Configure** in the Keystore section of the page.

   The Keystore Configuration page is displayed, as shown in Figure 10–16.

Figure 10–16 Keystore Configuration



Information
All changes made in this page require a server restart to take effect.

Keystore Configuration

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. You will need to provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Java Key Store (JKS)

Access Attributes

* Keystore Path: ./default-keystore.jks
* Password
* Confirm Password

Identity Certificates

Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore will use these identity certificates.

Signature Key

* Key Alias
* Signature Password
* Confirm Password

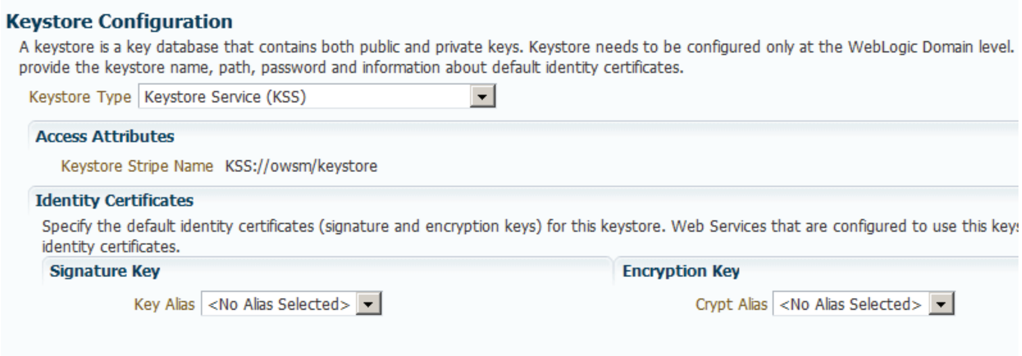
Encryption Key

* Crypt Alias
* Crypt Password
* Confirm Password

8. In the **Keystore Type** field, select `Keystore Service (KSS)` as the type.

   The KSS configuration page appears, as shown in Figure 10–17.

Figure 10–17 KSS Keystore Configuration Page



Keystore Configuration

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Keystore Service (KSS)

Access Attributes

Keystore Stripe Name: KSS://owsm/keystore

Identity Certificates

Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keys identity certificates.

Signature Key

Key Alias: <No Alias Selected>

Encryption Key

Crypt Alias: <No Alias Selected>

9. In the Identity Certificates section of the page, enter the alias for the signature and encryption keys as follows:

   - For the Signature Key, enter the alias name in the **Key Alias** field. The value you specify here must match the value in the keystore. For example, `orakey`.
   - For the Encryption Key, enter the alias name in the **Crypt Alias** field. The value you specify here must match the value in the keystore. For example, `orakey`.

   The alias for the signature and encryption keys is used to store and retrieve the keys.

Note: As described in "Configuring the Oracle WSM Keystore" on page 10-11, when you use Fusion Middleware Control to configure the Oracle WSM keystore, Oracle WSM creates entries in the credential store for the credential map `oracle.wsm.security`, and any keys that you define. No action on your part is required.

This is true for the KSS keystore as well, with one notable difference: the created `keystore.sig.csf.key` and `keystore.enc.csf.key` properties in the credential store point directly to the alias. (For JKS, these properties point to a CSF key that points to the sign key and password.)

10. Click **OK** to submit the changes.

    If you used a file-based keystore provider, the changes require a server restart to take effect. A restart is not required for database or LDAP-based keystore service providers.

11. Optionally, obtain trusted certificates, as described in "Importing a Certificate or Trusted Certificate with Fusion Middleware Control" in *Oracle Fusion Middleware Application Security Guide*.

12. Set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16.

10.6 Configuring Keystores for SSL

If you want to use any of the policies listed in "Which Policies Require You to Configure SSL?" on page 10-36 or "Which Policies Require You to Configure Two-Way SSL?" on page 10-37, you must configure keystores for SSL.

SSL provides secure connections by allowing two applications connecting over a network to authenticate the other's identity and by encrypting the data exchanged between the applications.

Authentication allows a server, and optionally a client, to verify the identity of the application on the other end of a network connection. Encryption makes data transmitted over the network intelligible only to the intended recipient. A client certificate (two-way SSL) can be used to authenticate the user.

This section describes how to set up a Web service client and the WebLogic Server Web service container to send requests over SSL.

To use SSL in a Web service application, you need to:

- Configure the WebLogic Server keystore and SSL settings.
- Configure the Web service client keystore and SSL settings.

These steps are described in the sections that follow.

10.6.1 Which Policies Require You to Configure SSL?

The predefined policies that require you to configure SSL are as follows:

- oracle/wss_http_token_over_ssl_service_policy
- oracle/wss_http_token_over_ssl_client_policy
- oracle/wss_saml_token_bearer_over_ssl_server_policy

- oracle/wss_saml_token_bearer_over_ssl_client_policy
- oracle/wss_saml_token_over_ssl_service_policy
- oracle/wss_saml_token_over_ssl_client_policy
- oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template
- oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template
- oracle/wss_username_token_over_ssl_service_policy
- oracle/wss_username_token_over_ssl_client_policy

In addition, you can create a new policy that requires SSL by using the following templates:

- oracle/wss_http_token_over_ssl_service_template
- oracle/wss_http_token_over_ssl_client_template
- oracle/wss_saml_token_bearer_over_ssl_service_template
- oracle/wss_saml_token_bearer_over_ssl_client_template
- oracle/wss_saml_token_over_ssl_service_template
- oracle/wss_saml_token_over_ssl_client_template
- oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template
- oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template
- oracle/wss_username_token_over_ssl_service_template
- oracle/wss_username_token_over_ssl_client_template

See Appendix C, "Predefined Assertion Templates" and Appendix B, "Predefined Policies" for more information on these assertions and policies.

10.6.2 Which Policies Require You to Configure Two-Way SSL?

The predefined policies that require you to configure two-way SSL are as follows:

- oracle/wss_saml_token_over_ssl_client_policy
- oracle/wss_saml_token_over_ssl_service_policy
- oracle/wss_username_token_over_ssl_client_policy, when mutual authentication is selected.
- oracle/wss_username_token_over_ssl_service_policy, when mutual authentication is selected.
- oracle/wss_http_token_over_ssl_client_policy, when mutual authentication is selected.
- oracle/wss_http_token_over_ssl_service_policy, when mutual authentication is selected.

In addition, you can create a new policy that requires two-way SSL by using the following templates:

- oracle/wss_saml_token_over_ssl_client_template
- oracle/wss_saml_token_over_ssl_service_template

10.6.3 How to Configure a Keystore on WebLogic Server

Private keys, digital certificates, and trusted certificate authority certificates establish and verify identity and trust in the WebLogic Server environment.

This section briefly summarizes the steps that are required to configure the keystore in WebLogic Server. See the following two sources for complete information:

- *Oracle WebLogic Server Administration Console Help* for complete information, particularly the topic "Servers: Configuration: Keystores."
- *Securing Oracle WebLogic Server*, particularly *Configuring Identity and Trust*.

WebLogic Server is configured with a default identity keystore *DemoIdentity.jks* and a default trust keystore *DemoTrust.jks*. In addition, WebLogic Server trusts the certificate authorities in the cacerts file in the JDK. This default keystore configuration is appropriate for testing and development purposes. However, these keystores should not be used in a production environment.

To configure identity and trust for a server:

1. Obtain digital certificates, private keys, and trusted CA certificates from the keytool utility, or a reputable vendor such as Entrust or Verisign, and include them in the keystore.

   To get the certificate, you must create a Certificate Request and submit it to the CA. The CA will authenticate the certificate requestor and create a digital certificate based on the request.

   The PEM (Privacy Enhanced Mail) format is the preferred format for private keys, digital certificates, and trusted certificate authorities (CAs).

   If you use the keytool utility, the default key pair generation algorithm is Digital Signature Algorithm (DSA). WebLogic Server does not support DSA. Specify another key pair generation and signature algorithm such as RSA when using WebLogic Server. For more information about the keytool utility, see the keytool-Key and Certificate Management Tool description at `http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html`.

   You can also use the digital certificates, private keys, and trusted CA certificates provided by the WebLogic Server kit. The demonstration digital certificates, private keys, and trusted CA certificates should be used only in a development environment.

2. Create one keystore for identity and one for trust. The preferred keystore format is JKS (Java KeyStore).
3. Load the private keys and trusted CAs into the keystores.
4. In the left pane of the Console, expand Environment and select **Servers**.
5. Click the name of the server for which you want to configure the identity and trust keystores.
6. Select **Configuration**, and then **Keystores**.
7. In the Keystores field, select the method for storing and managing private keys/digital certificate pairs and trusted CA certificates. These options are available:

   - Custom Identity and Custom Trust: Identity and trust keystores you create.

- Demo Identity and Demo Trust: The demonstration identity and trust keystores, located in the *..\server\lib* directory and the JDK cacerts keystore, are configured by default. Use for development only.
- Custom Identity and Java Standard Trust: A keystore you create and the trusted CAs defined in the cacerts file in the *JAVA_HOME\jre\lib\security* directory.
- Custom Identity and Command Line Trust: An identity keystore you create and command-line arguments that specify the location of the trust keystore.

8. In the Identity section, define attributes for the identity keystore.
   - Custom Identity Keystore: The fully qualified path to the identity keystore.
   - Custom Identity Keystore Type: The type of the keystore. Generally, this attribute is Java KeyStore (JKS); if left blank, it defaults to JKS.
   - Custom Identity Keystore Passphrase: The password you will enter when reading or writing to the keystore. This attribute is optional or required depending on the type of keystore. All keystores require the passphrase to write to the keystore. However, some keystores do not require the passphrase to read from the keystore. WebLogic Server only reads from the keystore so whether or not you define this property depends on the requirements of the keystore.

   Note: The passphrase for the Demo Identity keystore is *DemoIdentityKeyStorePassPhrase*.

9. In the Trust section, define properties for the trust keystore.

   If you chose Java Standard Trust as your keystore, specify the password defined when creating the keystore. Confirm the password.

   If you chose Custom Trust, define the following attributes:
   - Custom Trust Keystore: The fully qualified path to the trust keystore.
   - Custom Trust Keystore Type: The type of the keystore. Generally, this attribute is JKS; if left blank, it defaults to JKS.
   - Custom Trust Keystore Passphrase: The password you will enter when reading or writing to the keystore. This attribute is optional or required depending on the type of keystore. All keystores require the passphrase to write to the keystore. However, some keystores do not require the passphrase to read from the keystore. WebLogic Server only reads from the keystore, so whether or not you define this property depends on the requirements of the keystore.
10. The changes are automatically activated.

10.6.4 Configuring SSL on WebLogic Server (One-Way)

With one-way SSL, the server is required to present a certificate to the client but the client is not required to present a certificate to the server.

After you configure identity and trust keystores for a WebLogic Server instance as described in "Configuring Keystores for SSL" on page 10-36, you configure its SSL attributes. These attributes describe the location of the identity key and certificate in the keystore specified on the Configuration: Keystores page. Use the Configuration: SSL page to specify this information.

This section summarizes the steps required to configure SSL on WebLogic Server. For complete information, see *Securing Oracle WebLogic Server*.

To configure SSL:

1. In the left pane of the WebLogic Server Administration Console, expand Environment and select **Servers**.
2. Click the name of the server for which you want to configure SSL.
3. Select **Configuration**, and then the **SSL** page, and choose the location of identity (certificate and private key) and trust (trusted CAs) for WebLogic Server.
4. Set SSL attributes for the private key alias and password.
5. At the bottom of the page, click **Advanced**.
6. Set Hostname Verification to None.
7. Indicate the number of times WebLogic Server can use an exportable key between a domestic server and an exportable client before generating a new key. The more secure you want WebLogic Server to be, the fewer times the key should be used before generating a new key.
8. Set the Two Way Client Cert Behavior control to Client Certs Not Requested.
9. Specify the inbound and outbound SSL certificate validation methods. These options are available:
   - Builtin SSL Validation Only: Uses the built-in trusted CA-based validation. This is the default.
   - Built-in SSL Validation and Cert Path Validators: Uses the built-in trusted CA-based validation and uses configured CertPathValidator providers to perform extra validation.

10.6.5 Configuring SSL on WebLogic Server (Two-Way)

With two-way SSL, the server presents a certificate to the client and the client presents a certificate to the server. WebLogic Server can be configured to require clients to submit valid and trusted certificates before completing the SSL handshake.

After you configure identity and trust keystores for a WebLogic Server instance as described in "Configuring Keystores for SSL" on page 10-36, you can configure its two-way SSL attributes if the policy or template you are using requires it, as described in "Which Policies Require You to Configure Two-Way SSL?" on page 10-37.

This section summarizes the steps required to configure SSL on WebLogic Server. For complete information, see *Securing Oracle WebLogic Server*.

To configure two-way SSL:

1. In the left pane of the WebLogic Server Administration Console, expand Environment and select **Servers**.
2. Click the name of the server for which you want to configure SSL.
3. Select **Configuration**, and then the **SSL** page, and choose the location of identity (certificate and private key) and trust (trusted CAs) for WebLogic Server.
4. Set SSL attributes for the private key alias and password.
5. At the bottom of the page, click **Advanced**.

6. Set Hostname Verification to None.
7. Indicate the number of times WebLogic Server can use an exportable key between a domestic server and an exportable client before generating a new key. The more secure you want WebLogic Server to be, the fewer times the key should be used before generating a new key.
8. Set the Use Server Certs control if needed. Setting this control determines whether a Web service client hosted on WebLogic Server should use the server certificates/key as the client identity when initiating a connection over HTTPS.
9. Set the **Two Way Client Cert Behavior** control to Client Certs Requested and Enforced.
10. Specify the inbound and outbound SSL certificate validation methods. These options are available:
    - Builtin SSL Validation Only: Uses the built-in trusted CA-based validation. This is the default.
    - Builtin SSL Validation and Cert Path Validators: Uses the built-in trusted CA-based validation and uses configured CertPathValidator providers to perform extra validation.

10.6.6 Configuring SSL for a Web Service Client

The core WebLogic Server security subsystem uses private key and X.509 certificate pairs, stored in the default keystores, for SSL.

You must ensure that the Web service client trusts the X.509 certificate that WebLogic Server uses to digitally sign the request. Do one of the following:

1. Ensure that WebLogic Server obtains a digital certificate that the client automatically trusts, because it has been issued by a trusted certificate authority.
2. Create a certificate registry that lists all the individual certificates trusted by WebLogic Server, and then ensure that the client trusts these registered certificates.

To configure SSL for a Web service client:

1. Create a keystore used by the client application. Oracle recommends that you create one client keystore per application user.

   You can use the keytool utility to perform this step. For development purposes, the keytool utility is the easiest way to get started.
2. Create a private key and digital certificate pair, and load it into the client keystore.

   Make sure that the certificate's key usage allows both encryption and digital signatures.   Oracle requires a key length of 1024 bits or larger.
3. Make sure that the following properties are set in the client's JVM:
    - javax.net.ssl.trustStore -- The name of the file that contains the trust store.
    - javax.net.ssl.trustStoreType -- The type of KeyStore object that you want the default TrustManager to use.
    - javax.net.ssl.trustStorePassword -- The password for the KeyStore object that you want the default TrustManager to use.

10.6.7 Configuring Two-Way SSL for a Web Service Client

> **Note:** See "Configuring SOA Composite Applications for Two-Way SSL Communication" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite* for specific configuration steps when a SOA application is the Web service client over two-way SSL.

You must ensure that WebLogic Server is able to validate the X.509 certificate that the client uses to digitally sign its request, and that WebLogic Server in turn uses to encrypt its responses to the client. Do one of the following:

1. Ensure that the client application obtains a digital certificate that WebLogic Server automatically trusts, because it has been issued by a trusted certificate authority.
2. Create a certificate registry that lists all the individual certificates trusted by WebLogic Server, and then ensure that the client uses one of these registered certificates.

To configure SSL for a Web service client:

1. Create a keystore used by the client application. Oracle recommends that you create one client keystore per application user.

   You can use the keytool utility to perform this step. For development purposes, the keytool utility is the easiest way to get started.

2. Create a private key and digital certificate pair, and load it into the client keystore.

   Make sure that the certificate's key usage allows both encryption and digital signatures. Oracle requires a key length of 1024 bits or larger.

3. Make sure that the following properties are set in the client's JVM:

   - javax.net.ssl.trustStore -- The name of the file that contains the trust store.
   - javax.net.ssl.trustStoreType -- The type of KeyStore object that you want the default TrustManager to use.
   - javax.net.ssl.trustStorePassword -- The password for the KeyStore object that you want the default TrustManager to use.
   - javax.net.ssl.keyStore -- The name of the file that contains the KeyStore object.
   - javax.net.ssl.keyStoreType -- The type of KeyStore object.
   - javax.net.ssl.keyStorePassword -- The password for the KeyStore.

10.6.8 Configuring Synchronization of JKS Keystore File on Cluster

If you configure JKS keystore in fresh installs or use JKS in upgrade scenarios and want synchronization of JKS keystore file to happen on cluster without server restart, then perform the following steps:

1. Open the wsm-client-mbeans.xml file in a text editor.
2. Add the following property manually to wsm-client-mbeans.xml:

   ```
   <config-file path="../default-keystore.jks"/>
   ```

Note: The default-keystore.jks can have other name as well. Therefore, the file path must be provided accordingly.

JKS MBean in the wsm-client-mbeans.xml file should be similar to:

```
<?xml version = '1.0' encoding = 'UTF-8' standalone='yes'?>
<application-mbeans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/11/appli
cation-mbeans-11_1.xsd" schema-major-version="11" schema-minor-version="1">
<config-mbeans>
      <jmx-config-mbean
             objectname="oracle.wsm:type=security,name=JKSKeystoreMBean"
             class="oracle.wsm.security.store.jks.mgmt.KeystoreMBeanImpl"

 management-interface="oracle.wsm.security.store.jks.mgmt.KeystoreMBean">
            <description>MBean to access and manage JKS
Keystore</description>
            <config-file path="../default-keystore.jks"/>
        </jmx-config-mbean>
    </config-mbeans>
</application-mbeans>
```

3. Save the wsm-client-mbeans.xml file.
4. After adding the config file path, restart the server.

10.7 Configuring SSL on Oracle HTTP Server

The HTTPS protocol uses an industry standard protocol called Secure Sockets Layer (SSL) to establish secure connections between clients and servers. You can use the HTTPS/SSL support offered by the Oracle HTTP Server as one of the communication protocols to communicate between the client and the Web service. This section describes how to set up a Web service client and a Web service using Oracle WSM policies to send requests over SSL. Oracle HTTP Server is configured as a Web proxy that intermediates between the client and Oracle WebLogic Server. SSL is enabled at Oracle HTTP Server and SSL transport is turned on between the client and Oracle HTTP Server. Communication remains non-SSL between Oracle HTTP Server and WebLogic Server. This section describes how to configure the policies that require one-way SSL and two-way SSL.

For more information, see:

- "Configuring SSL in Oracle Fusion Middleware", in *Oracle Fusion Middleware Administrator's Guide*
- "Configuring SSL" in *Securing Oracle WebLogic Server*
- "Set Up SSL" in the *Oracle WebLogic Server Administration Console Help*
- "Configuring Secure Sockets Layer" in *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*

10.7.1 One-Way SSL

For more information on the Oracle WSM policies that require one-way SSL configuration, see "Which Policies Require You to Configure SSL?" on page 10-36.

To use one-way SSL, you need to:

1. Configure the Oracle HTTP Server as follows:

   a. In the file *ORACLE_INSTANCE*/config/OHS/<ohs_name>/ssl.conf, configure Oracle HTTP Server as a Web proxy and specify the list of URLs you want to access, as shown in Example 10–1.

Example 10–1 Specifying URLs in ssl.conf

```
# added properties for configuring OHS as webproxy
<IfModule weblogic_module>
WebLogicHost <host>
WebLogicPort <port>
SecureProxy Off
WlProxySSL On
Debug ALL
WlLogFile /tmp/weblogic.log
#the location attributes list the urls you want to access via OHS
<Location
 /myWlsService>
        SetHandler weblogic-handler
        WebLogicHost <host>
        WeblogicPort <port>
</Location>
```

   b. In the same file, set the following properties under virtual host configuration to ensure the client certificate information is sent to WebLogic Server:

   ```
   SSLVerifyClient optional
   ```

   c. By default, SSL in enabled on Oracle HTTP Server. The default https port is 4443. For more information on configuring this port, see "Configuring SSL in Oracle Fusion Middleware" in *Oracle Fusion Middleware Administrator's Guide*.

   d. Restart Oracle HTTP Server.

   For more information, see "Configuring SSL in Oracle Fusion Middleware" in *Oracle Fusion Middleware Administrator's Guide*.

2. Create a wallet as described at "Managing Keystores, Wallets, and Certificates" in *Oracle Fusion Middleware Administrator's Guide* and replace the default wallet. The default wallet is located in the *ORACLE_INSTANCE*/config/OHS/<ohs_name>/keystores/default directory. See Example 10–2 for sample commands for creating a wallet.

Example 10–2 Sample Commands for One-Way SSL

```
./orapki wallet create -wallet <wallet_location> -pwd password -auto_login
./orapki wallet display -wallet <wallet_location> -pwd password
./orapki cert display -cert <wallet_location>/ohs.crt

./orapki wallet add -wallet <wallet_location> -keysize 512 -dn "CN=<host_
name>,OU=st,O=Oracle WSM,L=N,ST=delhi,C=IN"
-self_signed -validity 700 -serial_num 20 -cert <wallet_location>/ohs.crt -user_
cert -pwd password

./orapki wallet display -wallet <wallet_location> -pwd password

JAVA_HOME/bin/keytool -import -trustcacerts -file ohs.crt -alias sslcert -keystore
client_keystore.jks -storepass password
```

3. In the Oracle WebLogic Administration Console, perform the following:

a. Navigate to the Servers page in the Environment tab.

b. Click Adminserver and in Configuration, select General.

c. In the Advanced section, check the following: WebLogic Plug-In Enabled, and Client Cert Proxy Enabled.

d. Save the changes.

e. Set the same parameters for the SOA server.

   For more information, see "Server: Configuration: General" in the *Oracle WebLogic Server Administration Console Help*.

To modify the client to use one-way (server authentication mode), create a JSE client from the Web service using JDeveloper. Modify the parameters and properties as described in Example 10–3.

Example 10–3 JSE Client Using SSL

```
public static void main(String [] args)
  {
    class1Service = new Class1Service();
        SecurityPolicyFeature[] securityFeatures =
            new SecurityPolicyFeature[] { new SecurityPolicyFeature("oracle/wss_
saml_token_over_ssl_client_policy") };
    Class1 class1 = class1Service.getClass1Port(securityFeatures);
    ((BindingProvider) class1).getRequestContext().put(BindingProvider.ENDPOINT_
ADDRESS_PROPERTY,
        "https://<host>:4443/myWlsService/Class1Port");

    ((BindingProvider) class1).getRequestContext().put(BindingProvider.USERNAME_
PROPERTY, "weblogic");
    System.setProperty("javax.net.ssl.trustStore","D:\\Oracle WSM_
QA\\11g\\PS2\\OHS\\wallet\\client_keystore.jks");
    System.setProperty("javax.net.ssl.trustStorePassword","password");
    System.setProperty("javax.net.ssl.trustStoreType","JKS");

    System.setProperty("weblogic.security.SSL.ignoreHostnameVerification" ,
 "true");
    System.setProperty("java.protocol.handler.pkgs",
 "com.sun.net.ssl.internal.www.protocol");
    System.setProperty("javax.net.debug","all");

    System.out.println("Call to the SSL service...");
    String response1 = class1.sayHello("test");
    System.out.println("Response = " + response1);
  }
```

10.7.2 Two-Way SSL

For more information on the Oracle WSM policies that require two-way SSL configuration, see "Which Policies Require You to Configure Two-Way SSL?" on page 10-37.

To use two-way SSL, you need to:

1. Configure the Oracle HTTP Server as follows:

   a. In the file *ORACLE_INSTANCE*`/config/OHS/<ohs_name>/ssl.conf`, configure Oracle HTTP Server as a Web proxy and specify the list of URLs you want to access as shown in Example 10–4.

Example 10–4 Specifying URLs in ssl.conf

```
 # added properties for configuring OHS as webproxy
<IfModule weblogic_module>
WebLogicHost <host>
WebLogicPort <port>
SecureProxy Off
WlProxySSL On
Debug ALL
WlLogFile /tmp/weblogic.log
#the location attributes list the urls you want to access via OHS
<Location /myWlsService>
        SetHandler weblogic-handler
        WebLogicHost <host>
        WeblogicPort <port>
</Location>
```

   b. In the same file, set the following properties under virtual host configuration to ensure the client certificate information is sent to the WebLogic Server:

   `SSLVerifyClient optional`

   `SSLOptions +StdEnvVars +ExportCertData`

   `SSLOptions +ExportCertData` is a mod_ssl directive that ensures certificate-related information is sent to WebLogic Server. `SSLOptions +StdEnvVars +ExportCertData` ensures that SSL-related information is sent.

   c. By default, SSL in enabled on Oracle HTTP Server. The default https port is 4443. For more information on configuring this port, see "Configuring SSL in Oracle Fusion Middleware" in *Oracle Fusion Middleware Administrator's Guide*.

   d. Restart Oracle HTTP Server.

   For more information, see "Configuring SSL in Oracle Fusion Middleware" in *Oracle Fusion Middleware Administrator's Guide*.

2. Create a wallet as described at "Managing Keystores, Wallets, and Certificates" in *Oracle Fusion Middleware Administrator's Guide* and replace the default wallet. The default wallet is located in the *ORACLE_INSTANCE*`/config/OHS/<ohs_name>/keystores/default` directory. See Example 10–5 for sample commands.

Example 10–5 Sample Commands for Two-Way SSL

```
JAVA_HOME/bin/keytool -genkey -alias twowayssl -keyalg RSA
 -keystore twowaykeystore.jks -storepass password -validity 700
./orapki wallet add -wallet <wallet_location> -cert
 <wallet_location>/twowayssl.crt -trusted_cert -pwd password
```

3. In the Oracle WebLogic Administration Console, perform the following:

   a. Navigate to the Servers page in the Environment tab.

   b. Click Adminserver and in Configuration, select General.

   c. In the Advanced section, check the following: WebLogic Plug-In Enabled, and Client Cert Proxy Enabled.

   d. Save the changes.

   e. Set the same parameters for the SOA server.

   For more information, see "Server: Configuration: General" in the *Oracle WebLogic Server Administration Console Help*.

To modify the client to use two-way (mutual authentication mode) SSL, create a JSE client from the Web service using JDeveloper. Modify the parameters and properties as described in Example 10–6.

Example 10–6 JSE Client Using SSL

```
public static void main(String [] args)
  {
    class1Service = new Class1Service();
        SecurityPolicyFeature[] securityFeatures =
            new SecurityPolicyFeature[] { new SecurityPolicyFeature("oracle/wss_
username_token_over_ssl_client_policy") };
    Class1 class1 = class1Service.getClass1Port(securityFeatures);
    ((BindingProvider) class1).getRequestContext().put(BindingProvider.ENDPOINT_
ADDRESS_PROPERTY,
        "https://<host>:4443/myWlsService/Class1Port");

    ((BindingProvider) class1).getRequestContext().put(BindingProvider.USERNAME_
PROPERTY, "weblogic");
    ((BindingProvider) class1).getRequestContext().put(BindingProvider.PASSWORD_
PROPERTY, "password");
    System.setProperty("javax.net.ssl.trustStore","D:\\Oracle WSM_
QA\\11g\\PS2\\OHS\\wallet\\twowaykeystore.jks");
    System.setProperty("javax.net.ssl.trustStorePassword","password");
    System.setProperty("javax.net.ssl.trustStoreType","JKS");
    System.setProperty("javax.net.ssl.keyStore","D:\\Oracle WSM_
QA\\11g\\PS2\\OHS\\wallet\\twowaykeystore.jks");
    System.setProperty("javax.net.ssl.keyStorePassword","password");
    System.setProperty("javax.net.ssl.keyStoreType","JKS");

    System.setProperty("weblogic.security.SSL.ignoreHostnameVerification" ,
 "true");
    System.setProperty("java.protocol.handler.pkgs",
 "com.sun.net.ssl.internal.www.protocol");
    System.setProperty("javax.net.debug","all");

    System.out.println("Call to the SSL service...");
    String response1 = class1.sayHello("test");
    System.out.println("Response = " + response1);
  }
```

10.8 Hardware Integration

This section describes setup information for the following hardware-related topics:

- "Using Hardware Security Modules With Oracle WSM" on page 10-47
- "Configuring Oracle WSM for Oracle SPARC T4 Cryptographic Acceleration" on page 10-51

10.8.1 Using Hardware Security Modules With Oracle WSM

Hardware security modules (HSM) are certified to operate with Oracle Advanced Security. These modules provide a secure way to store keys and off-load cryptographic processing.

10.8.1.1 Using SafeNet Luna SA With Oracle WSM for Key Storage

SafeNet Luna SA is a network-attached, (HSM featuring cryptographic processing and hardware key management for applications. Luna SA is designed to protect critical cryptographic keys across a wide range of security applications.

Some key advantages of using Luna SA with Oracle WSM are:

- Network shareability
- Most secure with keys always in hardware
- FIPS validated

> **Note:** You must contact your SafeNet representative to obtain certified hardware and software to use with Oracle Advanced Security.

By default, Oracle Web Services Manager (Oracle WSM) uses Java Key Store (JKS) for key storage. Keys and certificates required by Oracle WSM for cryptographic operations are fetched from a keystore file. When Luna SA is available in-network, it can be leveraged by Oracle WSM for key storage purposes and cryptographic operations.

This section includes the following topics:

- "About Installing and Configuring the Luna SA HSM Client" on page 10-48
- "Configuring the JRE Used By Oracle WSM" on page 10-49
- "Logging On to Luna SA" on page 10-49
- "Copying Keys and Certificates from JKS to Luna SA" on page 10-50
- "Configuring Oracle WSM to Use Luna SA" on page 10-50

10.8.1.2 About Installing and Configuring the Luna SA HSM Client

The Luna SA HSM client needs to be installed on the host that has a running instance of Oracle WSM. Then the Luna SA HSM client will communicate with an available Luna SA HSM network. However, this section does not cover Luna SA client installation, nor does it cover the Luna SA network installation and setup, which are out of scope for this document. Instead, you should refer to the Luna SA documentation for those instructions, at `http://www.safenet-inc.com/Products/Detail.aspx?id=2147483853&terms=search`.

Before you installing the Luna SA HSM client, verify the following checklist:

- You already have Luna SA installed and available in you network.
- You are logged in as root or as a user that has installation permission.
- You have a Luna SA client installation CD or software image.
- You have all required passwords for Luna SA, including an administrator password and a partition password.

> **Note:** You must contact your SafeNet representative to have the hardware security module, and to acquire the necessary library.
>
> These tasks must be performed before you can use an Luna SA hardware security module with Oracle WSM

10.8.1.3 Configuring the JRE Used By Oracle WSM

After installing the Luna SA client, you need to configure the JRE that will be used by the Oracle WSM setup.

1. Copy the following JAR files from the `/usr/lunasa/jsp/lib` directory to the `$JAVA_HOME/jre/lib/ext` directory:
   - `LunaJCASP.jar`
   - `LunaJCESP.jar`
2. Copy the `libLunaAPI.so` file to the `java.library.path`.
3. Edit the `$JAVA_HOME/jre/lib/security/java.security` file to include two Luna providers.

   At the end of the `security.providers` list add these two Luna providers:

   ```
   security.provider.n=com.chrysalisits.crypto.LunaJCAProvider
   security.provider.n+1=com.chrysalisits.cryptox.LunaJCEProvider
   ```

   where

   n specifies the preference order that determines the order in which providers are searched for requested algorithms when no specific provider is requested. The order is 1-based; 1 is the most preferred, followed by 2, and so on.

10.8.1.4 Logging On to Luna SA

Before you can use Luna SA with Oracle WSM, you must log on to the Luna SA server. This is one-time process that creates a Luna log-in session on the client machine. This session remains active until the client or server machine is rebooted, or when someone explicitly logs out of the Luna session.

You must use the `salogin` utility to log in. The `salogin` utility establishes a connection between the client and the HSM partition for a particular application. It takes an application ID as an argument. This application id consists of two parts: a high and a low ID.

Before invoking the `salogin` utility, you need to add an entry to the `Chrystoki.conf` file, which registers the application ID. The `Chrystoki.conf` file is usually found in the `/etc/` directory. This is also a one-time process.

1. Edit the `/etc/Chrystoki.conf` file by adding the application ID to the end of file. For example:

   ```
   Misc = {
    AppIdMajor=<major id>;
    AppIdMinor=<minor id>;
    }
   ```

2. Log into the Luna SA server, by entering:

   ```
   /salogin -o -s <partition number> -i <AppIdMajor>:<AppIdMinor> -v -p
   <partition_password>
   ```

   This opens a session for the application ID you provided. The `salogin` is in the `/usr/lunasa/bin` directory.

3. To log out of the Luna SA server, enter:

   ```
   salogin -c -s <slot number> -i <AppIdMajor>:<AppIdMinor>
   ```

10.8.1.5 Copying Keys and Certificates from JKS to Luna SA

If keys and certificates are currently in the JKS, then you need to move all keys and certificates to LunaSA. You can use the `cmu` script provided by LunaSA for importing keys and certificates.

- The `cmu importKey` command imports an RSA|DSA private key from a file onto an HSM. (Supports PKCS12(RSA), PKCS8(RSA/DSA), or PKCS1(RSA)).
- The `cmu import` command imports an X.509 certificate from a file onto an HSM.

10.8.1.6 Configuring Oracle WSM to Use Luna SA

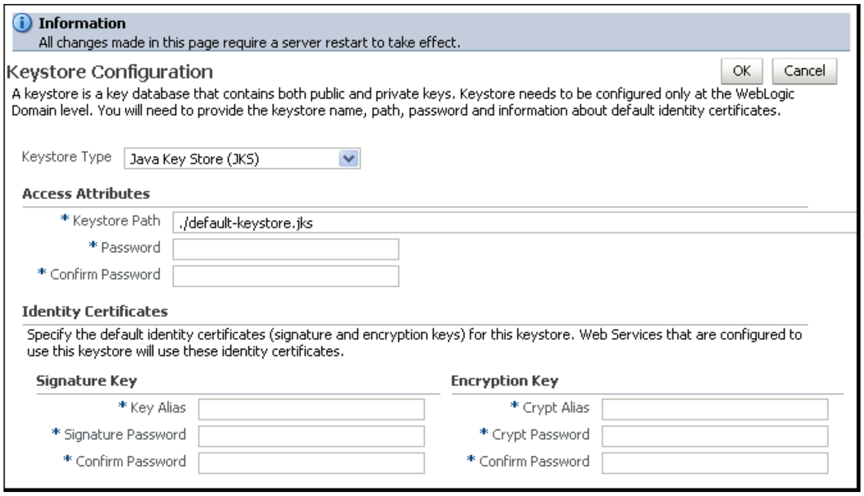
As part of configuring Oracle WSM to use Luna SA, the keystore type has to be changed to *Luna* from the default *Java Key Store (JKS)* value.

Follow these steps to configure the keystore type:

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the keystore. Select the domain.
2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Security**, and then **Security Provider Configuration**.
3. Click the plus sign (+) to expand the **Keystore** control near the bottom of the page, and then click **Configure**.

   The Web Services Manager Keystore Configuration page is displayed, as shown in Figure 10–18.

Figure 10–18 Web Services Manager Keystore Configuration Page



Information
All changes made in this page require a server restart to take effect.

Keystore Configuration

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. You will need to provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Java Key Store (JKS)

Access Attributes

* Keystore Path: ./default-keystore.jks
* Password:
* Confirm Password:

Identity Certificates

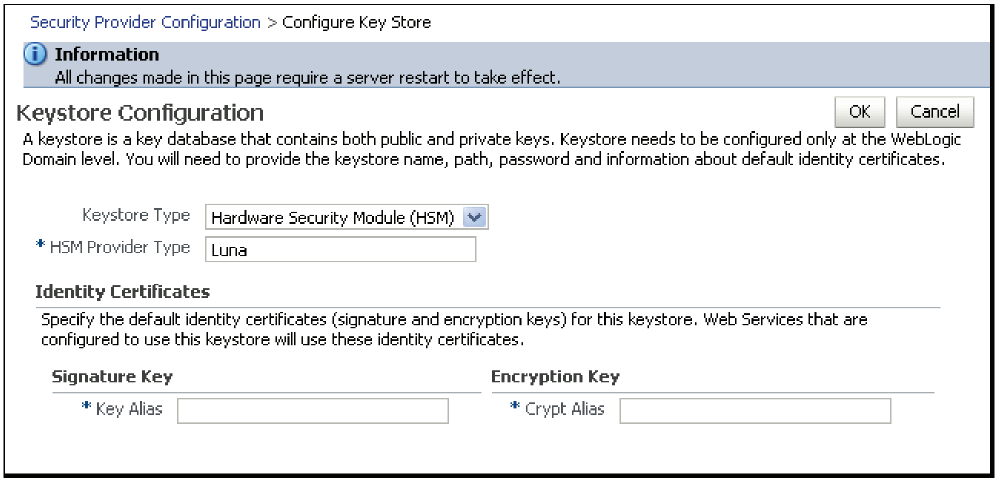
Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore will use these identity certificates.

Signature Key
* Key Alias:
* Signature Password:
* Confirm Password:

Encryption Key
* Crypt Alias:
* Crypt Password:
* Confirm Password:

4. In the **Keystore Type** drop-down, select **Hardware Security Module (HSM)**.
5. After the Keystore Configuration page refreshes, enter *Luna* in the **HSM Provider Type** field, as shown is Figure 10–19.

Figure 10–19 Web Services Manager Keystore Configuration Page (Refreshed)



Security Provider Configuration > Configure Key Store

Information
All changes made in this page require a server restart to take effect.

Keystore Configuration

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. You will need to provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Hardware Security Module (HSM)
* HSM Provider Type: Luna

Identity Certificates

Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore will use these identity certificates.

Signature Key
* Key Alias:

Encryption Key
* Crypt Alias:

6. In the **Key Alias** and **Crypt Alias** fields, enter an alias for the signature and encryption keys. (Note that Luna SA does not require passwords to access the keystore and private keys.)

   For HSMs, only a key alias is required so all `*csf.key` (`keystore.sig.csf.key` and `keystore.enc.csf.key`) properties should have a direct alias and not credential store keys. This information is also applicable to configuration overrides of `*csf.key` properties.

7. Click **OK** to submit the changes.

8. Restart Fusion Middleware Control.

10.8.2 Configuring Oracle WSM for Oracle SPARC T4 Cryptographic Acceleration

Oracle WSM supports the use of Oracle SPARC T4 processor-based servers, which eliminate the need for third-party security hardware by integrating computing, security, and I/O on a single chip. Deploying Oracle WSM on Oracle SPARC T4 based servers transparently leverages the T4 processor based cryptographic capabilities. This delivers high-performance security for scenarios that rely on compute-intensive cryptographic operations, such as those imposed by transport-layer and message-layer protection policies.

This section describes how to configure Oracle WSM to take advantage of cryptographic acceleration capabilities of Oracle SPARC T4 processor-based servers.

This section applies only to users who are running Oracle SPARC T4 processor-based servers running Oracle Solaris 10 8/11 or later.

The following topics are described:

- "Terms You Need to Understand" on page 10-52
- "Overview of Oracle SPARC T4 Hardware Assisted Cryptographic Acceleration" on page 10-52
- "Configuring Transport-Level Security for Cryptographic Acceleration" on page 10-53
- "Configuring Message-level Security for Cryptographic Acceleration" on page 10-54
- "Additional Reading" on page 10-56

10.8.2.1 Terms You Need to Understand

This section uses the following terms that you need to understand. Refer to the whitepaper described in "Additional Reading" on page 10-56 for a complete discussion of these terms.

- **PKCS#11 token** — A token that generically refers to all the hardware and software tokens that implement the PKCS#11 API. The PKCS#11 API is an RSA standard for integrating hardware cryptographic accelerators, cryptographic tokens (for example, SCA-6000), and smart cards.

  A software based PKCS#11 token is a PKCS#11 token implemented entirely in software (for example, Solaris PKCS11 Softtoken.)

- **Solaris Cryptographic Framework** — The Solaris Cryptographic Framework (SCF) library plays a vital role in providing application access to hardware-assisted cryptographic acceleration provided by Oracle T-series processors and Hardware Security Modules (HSM), including the Oracle Sun Crypto Accelerator 6000 PCIe Card (SCA-6000) and third-party HSMs. SCF is based on PKCS#11 standard interfaces and provides a set of cryptographic services for kernel-level and user-level consumers to perform cryptographic operations.

10.8.2.2 Overview of Oracle SPARC T4 Hardware Assisted Cryptographic Acceleration

The Oracle SPARC T4 processor is part of Oracle's SPARC T-series processors family, which combines multiprocessing at the processor core level and hardware multithreading inside of each core with an efficient instruction pipeline to enable Chip Level Multi-Threading (CMT). These processors present a unique "System-on-a-Chip" design principle that incorporates specialized features such as on-chip/on-core cryptographic acceleration, 10 Gigabit Ethernet networking, and hardware-enabled virtualization capabilities. Each core of the Oracle SPARC T4 processor contains a Stream Processing Unit (SPU) to perform processing of cryptographic operations at the same clock speed as the core. The SPU is designed to achieve wire-speed encryption and decryption on the processor's 10 GbE ports.

Configuring and deploying Oracle WSM on Oracle SPARC T4 based servers delivers high performance by leveraging the T4 on-core cryptographic instructions to perform computationally intensive cryptographic operations as part of Web service security transactions using SSL and WS-Security mechanisms. For example, all message protection policies are computationally intensive.

Oracle WSM makes use of SPARC T4 processor based cryptographic acceleration in the following scenarios:

- Transport-level security, as described in "Configuring Transport-Level Security for Cryptographic Acceleration" on page 10-53.

  Oracle WSM and WebLogic Server rely on the underlying Java Cryptographic Extensions (JCE) provider for supporting SSLv3/TLSv1 based secure communication. On Oracle Solaris/SPARC-based deployments, the Sun JCE provider is bundled with the Java runtime environment.

  The Java PKCS#11 interfaces off-load and accelerate the compute-intensive cryptographic workloads (for example, RSA, AES and ECC) of SSL/TLS protocols by using the on-core cryptographic instructions of SPARC T4 processor.

- Message-level security, as described in "Configuring Message-level Security for Cryptographic Acceleration" on page 10-54.

Message-level security builds on cryptographic operations that support Web Services security standards such as WS-Security, WS-SecurityPolicy, and WS-Trust.

In particular, Web services security makes use of public-key encryption, digital signature (for example, RSA, DSA and ECC), bulk encryption (for example, AES, 3DES, and DES) and message digest (for example, SHA-1, SHA-2, and MD5) functions intended for supporting XML encryption, XML digital signature and related cryptographic operations.

Oracle WSM implements a dedicated PKCS#11 interface to delegate cryptographic operations (via SCF) to on-core cryptographic instructions of SPARC T4 processor.

10.8.2.3 Configuring Transport-Level Security for Cryptographic Acceleration

Perform the following tasks to configure cryptographic acceleration for transport-level security:

1. Configure the WebLogic Server keystore, as described in "Configure keystores" in the *Oracle WebLogic Server Administration Console Help*.

   Choose "Custom Identity and Java Standard Trust" and Java KeyStore (JKS).

2. Enable JSSE SSL for WebLogic Server.

   Using JSSE based SSL automatically leverages the SunPKCS11 provider implementation pre-configured with the Java runtime environment on Solaris SPARC.

   See "Enabling and Disabling the JSSE-Based SSL Implementation" in *Securing Oracle WebLogic Server* for the steps to follow.

3. Confirm the use of the SunPKCS11 provider.

   The SunPKCS11 provider is a Java based PKCS#11 implementation that integrates with underlying PKCS#11 implementations provided by the SCF and its exposed cryptographic providers.

   In a typical WebLogic server installation on Solaris, the Java runtime environment is pre-configured to make use of the SunPKCS11 provider.

   Therefore, make sure that the SunPKCS11 provider is listed as the first provider in the `$JAVA_HOME/jre/lib/security/java.security properties` file:

   ```
   security.provider.1=sun.security.pkcs11.SunPKCS11
   ${java.home}/lib/security/sunpkcs11-solaris.cfg
   ```

   The relevant portion of the default Solaris `$JAVA_HOME/jre/lib/security/java.security` file is shown in Example 10–7.

Example 10–7 Partial java.security File

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.pkcs11.SunPKCS11
${java.home}/lib/security/sunpkcs11-solaris.cfg
security.provider.2=sun.security.provider.Sun
security.provider.3=sun.security.rsa.SunRsaSign
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
```

```
security.provider.9=sun.security.smartcardio.SunPCSC
```

4. Restart WebLogic Server.
5. Verify that the SSL configuration is working.

10.8.2.4 Configuring Message-level Security for Cryptographic Acceleration

Perform the following tasks to configure cryptographic acceleration for message-level security:

1. From the Solaris command line, create a PKCS11 keystore.

   To create and initialize a PKCS11 keystore, use the pktool setpin command.

   When you specify `keystore=pkcs11`, the keystore defaults to "Sun Software PKCS#11 softtoken."

   If the softoken keystore has not yet been initialized, use "changeme" as the original passphrase.

   ```
   # pktool setpin keystore=pkcs11
   Enter token passphrase:
   Create new passphrase:
   Re-enter new passphrase:
   Passphrase changed.
   ```

2. Generate private keys for the keystore.

   ```
   # pktool genkeypair keystore=pkcs11 keytype=rsa keylen=1024 hash=sha1
   ```

3. Alternatively, you can choose to import the key from a Java keystore to the Solaris Softtoken keystore by using the following command.

   ```
   # keytool -importkeystore
   -srckeystore /opt/Oracle/Middleware/default-keystore.jks
   -destkeystore NONE -srcstoretype JKS
   -deststoretype PKCS11
   -srcstorepass changeme -deststorepass your-scfpassword
   ```

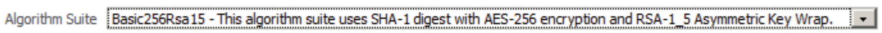
4. From the Solaris command line, verify that the keys are present in the Solaris Softtoken keystore.

   ```
   keytool -list -storetype pkcs11 -keystore NONE
   ```

5. Make sure that the algorithm suites of the policies you use are not in the disabledMechanisms list in the `$JAVA_HOME/jre/lib/security/sunpkcs11-solaris.cfgsunpkcs11-solaris.cfg` configuration file.

   For example, if the specified algorithm suite for a policy is Basic256Rsa15 as shown in Figure 10–20, it uses Aes256 encryption and KwAes256/kwRsA15 for key wrap. In this case, make sure that CKM_AES is not in the disabledMechanisms list in the configuration file.

Figure 10–20 Sample Algorithm Suite



Algorithm Suite: Basic256Rsa15 - This algorithm suite uses SHA-1 digest with AES-256 encryption and RSA-1_5 Asymmetric Key Wrap.

See Appendix A Sun PKCS#11 Provider's Supported Algorithms in the *Java PKCS#11 Reference Guide* for the list of supported algorithms.

6. Configure the Oracle WSM keystore.

   Follow these steps to configure the PKCS11 keystore type:

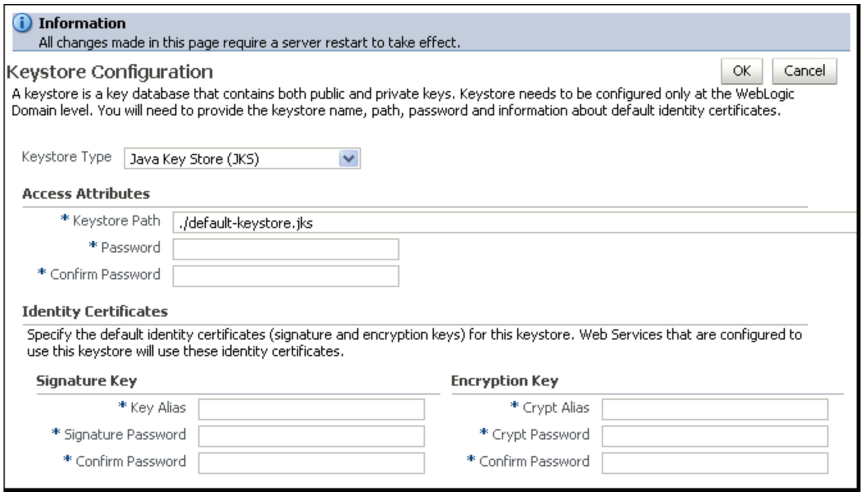
   a. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the keystore. Select the domain.

   b. Using Fusion Middleware Control, click **WebLogic Domain**, then **Security**, and then **Security Provider Configuration**.

   c. Click the plus sign (+) to expand the **Keystore** control near the bottom of the page, and then click **Configure**.

      The Web Services Manager Keystore Configuration page is displayed, as shown in Figure 10–18.

Figure 10–21 Web Services Manager Keystore Configuration Page



Information
All changes made in this page require a server restart to take effect.

Keystore Configuration

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. You will need to provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Java Key Store (JKS)

Access Attributes

* Keystore Path: ./default-keystore.jks
* Password
* Confirm Password

Identity Certificates

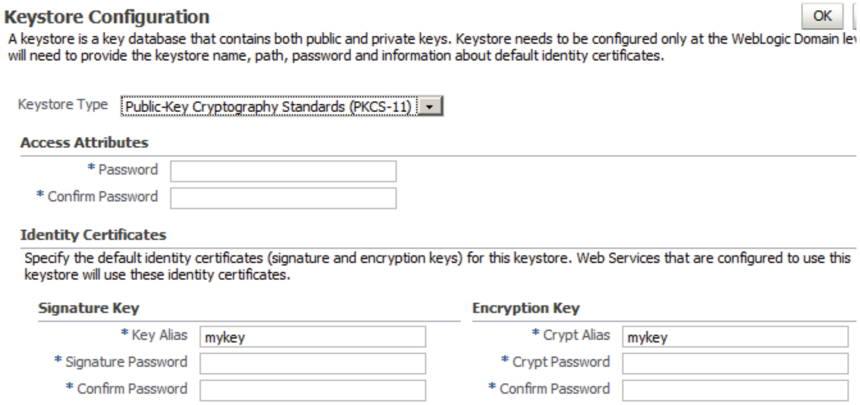
Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore will use these identity certificates.

Signature Key
* Key Alias
* Signature Password
* Confirm Password

Encryption Key
* Crypt Alias
* Crypt Password
* Confirm Password

   d. In the **Keystore Type** drop-down, select **Public Key Cryptographic Standards (PKCS-11)**, as shown is Figure 10–19.

Figure 10–22 Web Services Manager PKCS-11 Keystore



Keystore Configuration

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain le
will need to provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Public-Key Cryptography Standards (PKCS-11)

Access Attributes

* Password
* Confirm Password

Identity Certificates

Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore will use these identity certificates.

Signature Key
* Key Alias: mykey
* Signature Password
* Confirm Password

Encryption Key
* Crypt Alias: mykey
* Crypt Password
* Confirm Password

e. Enter the password for the keystore and confirm it. This password must match the PKCS11 keystore password you entered in Step 1.

f. In the **Key Alias** and **Crypt Alias** fields, enter an alias for the signature and encryption keys and the corresponding key passwords. The alias and password for the signature and encryption keys define the string alias and passwords used to store and retrieve these keys. These keys must be present in the PKCS11 keystore.

g. Click **OK** to submit the changes.

h. Restart Fusion Middleware Control.

7. Verify the configuration.

   To ensure the hardware-assisted cryptographic acceleration is correctly configured and working, use the following Solaris DTrace script.

```
#!/usr/sbin/dtrace -s
pid$1:libsoftcrypto:yf*:entry,
pid$1:libmd:yf*:entry
{
  @[probefunc] = count();
}
tick-10sec
{
  printa(@);
  clear(@);
  trunc(@,0);
}
tick-100sec
{exit(0);}
```

   Save this script as a file named `cryptoverify.d`. Run this script with the WebLogic Server's Java process ID as a command line argument, as follows:

```
# dtrace -s cryptoverify.d  <WeblogicServer Process ID>
```

   For example, in an encryption scenario using the AES algorithm, a positive and growing value of AES jobs indicates that cryptographic acceleration is operational on the target AES bulk encryption payloads.

10.8.2.5 Additional Reading

For information on deploying Oracle WSM on Oracle SPARC T2+/T3 processors, refer to the whitepaper *"High Performance Security for SOA and XML Web Services Using Oracle Web Services Manager and Oracle SPARC Enterprise T-series Servers,"* which is available at http://www.oracle.com/technetwork/articles/systems-hardware-architecture/hi-perf-soa-xml-svcs-172821.pdf.

The whitepaper is the definitive source for cryptographic acceleration information for using Oracle SPARC T2+ and T3 processor based servers

The whitepaper covers many additional pertinent topics such as Solaris Cryptographic Framework components, using Solaris Kernel SSL (KSSL), and performance characteristics.

10.9 Using Service Identity Certification Extension

For Web services that implement a message-protection policy, the Web service's base64-encoded public certificate is published in the WSDL. The certificate is included for message protection policies whether or not the policy encrypts or decrypts data.

> **Note:** In prior releases of Oracle WSM, for Web services that implemented a message-protection policy the Web service client needed to store the Web service's public certificate in its domain-level keystore. The client then used the `keystore.recipient.alias` property to identify the certificate in the keystore. To do this, you either identified the `keystore.recipient.alias` property on the Configurations page or overrode it on a per-client basis using the Security Configuration Details control when attaching the policy (or programmatically).

The certificate in the WSDL is the service's public key by default, as determined by the Encryption Key you specified when you configured the keystore as described in "Configuring Keystores for Message Protection" on page 10-9.

If this certificate is not found in the WSDL, the `keystore.recipient.alias` property is used instead and the certificate must be in the client's domain-level keystore as before.

> **Note:** Self-signed certificates must be available in the client-side keystore to be trusted.

10.9.1 Hostname Verification for the Certificate Included in WSDL

The hostname verification feature ensures that a certificate retrieved from a WSDL was not the subject of a substitution attack or "man in the middle" attack and is indeed the expected certificate.

To to this, Oracle WSM validates that the common name (CN) or the subject Group Base Distinguished Name (DN) in the certificate matches the hostname of the service.

This feature depends upon the subject DN of the certificate.

By default, hostname verification is disabled.

10.9.2 Enabling or Disabling Service Identity Certificate Extension and Hostname Verification

You use Fusion Middleware Control to enable or disable service identity certificate extension and hostname verification.

The properties on the Identity Extension tab enable you to specify whether to enforce Web service policies by publishing the X509 certificate in the WSDL. In addition, if the X509 is published, you can also specify whether to ignore hostname verification.

Service identity certificate extension is enabled by default; hostname verification is disabled by default.

Note: Service identity certificate extension does not set the encryption key from which the public key is derived. You must first specify this key as described in "Configuring Keystores for Message Protection" on page 10-9.

To enable or disable service identity certificate extension and hostname verification:

1. Set the encryption key from which the public key is derived, as described in "Configuring Keystores for Message Protection" on page 10-9.

   If you use a service side override to override the encryption key or keystore for a Web service, the certificate corresponding to the overridden key is used.

2. From the navigation pane, expand **WebLogic Domain**.
3. Select the domain in which you want to enable or disable service identity certificate extension and hostname verification.
4. Using Fusion Middleware Control, click **WebLogic Domain**.
5. Select **Web Services**, and then select **Platform Policy Configuration**.
6. Select the **Identity Extension** tab.
7. To modify a identity extension property, select it and then click **Edit**. In the Edit Property window, you can edit the Value field to change the default amount for each property.

   - `wsm.ignore.identity.wsdl` – Specifies whether to enable or disable the consumption of the X509 Certificate from a client-side WSDL, per domain. By default, this property is enabled (`false`), which means that the certificate from the WSDL will be used by the client run time for encryption. You can disable the consumption of the X509 Certificate by changing the default setting to `true`.
   - `wsm.ignore.hostname.verification` – Specifies whether to ignore the hostname verification feature per domain. By default this property is disabled (`true`). However, you can enable hostname verification by setting the property to `false`.

8. To delete an existing property, select it and then click **Delete**.
9. Click **Apply** to apply the property updates.

10.9.3 Ignoring the Service Identity Certificate Extension From the Client

Note: By default, if the certificate is published in the WSDL, then the client override property value for `keystore.recipient.alias` is ignored.

For a Java EE client, the value of the wsm.ignore.identity.wsdl property is read automatically and no additional configuration is required. Set this property in Fusion Middleware Control to turn identity verification on and off, as described in "Enabling or Disabling Service Identity Certificate Extension and Hostname Verification" on page 10-57.

For a JSE client, the Web service client must take explicit action to ignore the certificate in the WSDL and rely solely on the `keystore.recipient.alias` property it sets.

To do this, set the value of *wsm.ignore.identity.wsdl* to true:

```
BindingProvider.getRequestContext().put(SecurityConstants.ClientConstants.WSM_
IGNORE_IDENTITY_WSDL, "true");
```

10.9.4 Ignoring Hostname Verification from the Client

For a Java EE client, the value of the wsm.ignore.hostname.verification property is read automatically and no additional configuration is required. Set this property in Fusion Middleware Control to turn hostname verification on and off, as described in "Enabling or Disabling Service Identity Certificate Extension and Hostname Verification" on page 10-57.

For a JSE client, the Web service client must take explicit action to ignore hostname verification.

To do this, set the value of *wsm.ignore.hostname.verification* to true:

```
BindingProvider.getRequestContext().put(SecurityConstants.ClientConstants.WSM_
IGNORE_HOSTNAME_VERIFICATION,"false");
```

10.10 Configuring an Authentication Provider in WebLogic Server

This section introduces WebLogic Server security features that are described in detail in *Securing Oracle WebLogic Server* and in the *Oracle WebLogic Server Administration Console Help*. This section provides only a brief introduction to the security features, and concentrates on how they relate to configuring policies.

Policies that use any of the supported token types -- including username, X.509, Kerberos, SAML, HTTP BASIC and so forth -- require an authentication provider, such as the WebLogic Default Authentication provider.

The following policies fall into this category:

- oracle/wss_http_token_service_policy
- oracle/wss_username_token_service_policy
- oracle/wss_username_token_over_ssl_service_policy
- oracle/wss11_username_token_with_message_protection_service_policy
- oracle/wss10_username_token_with_message_protection_service_policy
- oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy
- oracle/wss10_x509_token_with_message_protection_service_policy
- oracle/wss10_saml_token_service_policy
- oracle/wss10_saml_token_with_message_protection_service_policy
- oracle/wss_saml_token_over_ssl
- oracle/wss_saml_token_bearer_over_ssl_service_policy
- oracle/wss10_saml_hok_token_with_message_protection_service_policy
- oracle/wss11_saml_token_with_message_protection_service_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy
- oracle/wss11_x509_token_with_message_protection_service_policy

10.10.1 What Type of WebLogic Security Authentication Providers Must You Create?

You can configure any of the WebLogic Server Authentication providers for use with Oracle WSM.   That is, you can use any of the following Web logic Server Authentication providers, regardless of the token type:

- The WebLogic Authentication provider, also known as the DefaultAuthenticator.
- Oracle Internet DirectoryAuthenticator or Oracle Virtual Directory Authenticator.
- LDAP Authenticator or Open LDAP Authenticator.
- RDBMS Authentication providers.

> **Note:** If you use an RDBMS authentication provider, or any other non-LDAP-based provider, there is a limitation that you cannot specify custom attributes to be added to the SAML assertion that Oracle WSM generates. This limitation does not exist for any of the LDAP-based providers.

This means that for policies that use SAML, Kerberos, and X.509 tokens, you do not have to configure a WebLogic Server provider to handle these specific token types.

More specifically, the Oracle WSM runtime does **not** use any other WebLogic Server providers, including but not limited to:

- Any Identity Assertion provider
- X509 providers

  Oracle WSM policies based on an X509 token do not use the WebLogic Server X509 Identity Assertion provider.
- SAML providers

  Oracle WSM policies based on a SAML token do not use the WebLogic Server SAML providers.
- Credential Mapper providers
- Authorization providers
- Role Mapper providers
- Certification Path provider
- Auditing provider

10.11 Configuring the SAML and Kerberos Login Modules

The SAML and Kerberos policies have associated login modules, as determined by the assertions that make up the policy. When you attach a SAML policy to a Web service, you can edit the login policy and make any needed changes.

You can configure the following SAML and Kerberos login modules:

- saml.loginmodule—The SAML login module is a Java Authentication and Authorization Service (JAAS) login module that accepts SAML assertions for a login. The SAML login module enables the Web services to run using the login context of the principal created from the SAML assertion.
- saml2.loginmodule—The SAML2 login module is a JAAS login module that accepts SAML2 assertions for a login. The SAML2 login module enables the Web

services to run using the login context of the principal created from the SAML2 assertion.

- krb5.loginmodule—The Kerberos login module is a JAAS login module that authenticates users using Kerberos protocols. The Kerberos login module has optional properties that you can configure.

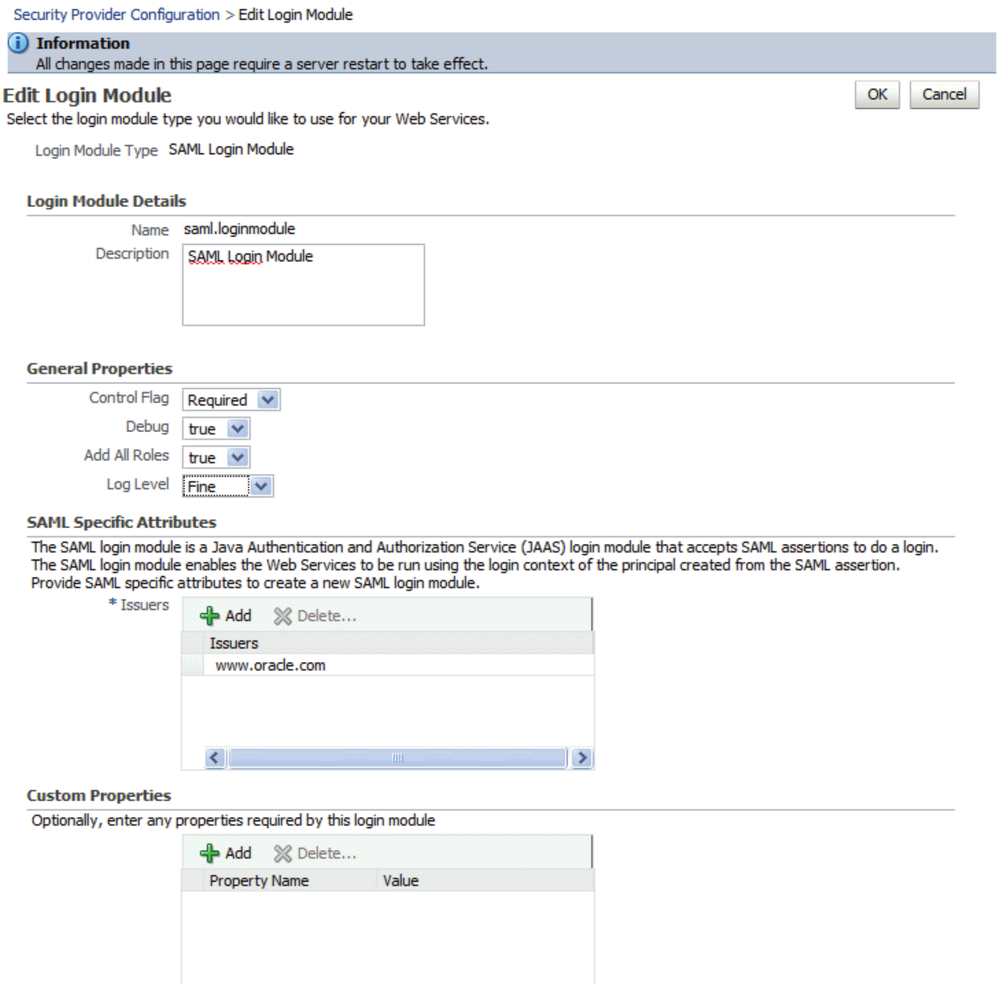
(Login modules associated with other policy types do not have settings specific to the Web service policies.)

To configure a login module:

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the login module. Select the domain.
2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Security**, and then **Security Provider Configuration**.
3. From the list of login modules, select a login module and click **Edit**.

   For example, if you select the saml.loginmodule from the list of login modules and click **Edit**, the Edit Login Module page shown in Figure 10–23is displayed.

Figure 10–23 Edit Login Module Page for SAML Login Module


Security Provider Configuration > Edit Login Module

Information
All changes made in this page require a server restart to take effect.

Edit Login Module
Select the login module type you would like to use for your Web Services.

Login Module Type SAML Login Module

Login Module Details

Name saml.loginmodule
Description SAML Login Module

General Properties

Control Flag Required
Debug true
Add All Roles true
Log Level Fine

SAML Specific Attributes

The SAML login module is a Java Authentication and Authorization Service (JAAS) login module that accepts SAML assertions to do a login.
The SAML login module enables the Web Services to be run using the login context of the principal created from the SAML assertion.
Provide SAML specific attributes to create a new SAML login module.

* Issuers

Issuers
www.oracle.com

Custom Properties

Optionally, enter any properties required by this login module

Property Name	Value

Note: Do not edit the default values in the General Properties section or unexpected results may occur. The default values for these properties are as follows:

- **Control Flag** —Required
- **Debug** — true
- **Add All Roles** — true
- **Log Level** — Fine

4. Optionally, in the SAML Specific Attributes section, configure an alternate Issuer attribute if required for your configuration. For SAML policies, the **Issuers** attribute is required. This attribute specifies the name of the issuer of the SAML or SAML2 token. For predefined Oracle SAML policies and assertions, the default value is `www.oracle.com`. If you are using the predefined SAML policies (or assertions) for both the Web service client and Web service sides, you can generally use the defaults and not configure any issuer. For more information, see "Adding an Additional SAML Assertion Issuer Name" on page 10-67.

5. In the Custom Properties section of the page, configure any custom properties for the login module.

   To add a property, click **Add** and enter a property name and value in the Add New Property window. Click **OK** to add the property to the Custom Properties list.

   To change the value of an existing property, you need to delete the property from the Custom Properties list and add a new property with the revised value.

Table 10–1 lists the SAML and Kerberos login modules and describes properties that you can configure.

Table 10–1 SAML and Kerberos Login Modules Attributes and Properties

Login Module Service Name	Property	Description
saml.loginmodule saml2.loginmodule	oracle.security.jps.assert.saml.identity	A domain-wide property used to determine the mapping between the SAML subject and the user. Valid values include: ▪ `false`—When this flag is set to `false`, the username in the SAML subject is mapped to the actual user in the identity store. The user roles and subject are created with username and roles specified in the identity store. This is the default. ▪ `true`—When this flag is set to `true`, the SAML subject is treated as a logical/virtual user. The user is not mapped to the actual user in the identity store. The subject is populated only with the username from the SAML subject. Because the subject is treated as a virtual user, identity store configuration is not required and the Authentication Provider is not invoked for all SAML policies in the domain using this login module.
	oracle.security.jps.add.assertion.to.subject	Boolean flag used to indicate whether the SAML assertion should be added to the authenticated subject as a private credential. The default is `true`.
krb5.loginmodule	principal	The name of the principal that should be used. It can be a simple username, such as "testuser", or a service name such as "host/testhost.eng.sun.com". You can use the principal option to set the principal when there are credentials for multiple principals in the keyTab or when you want a specific ticket cache only.
	useKeyTab	True or false. Set this to true if you want the module to get the principal's key from the keytab (default value is False). If keytab is not set, then the module will locate the keytab from the Kerberos configuration file. If it is not specified in the Kerberos configuration file then it will look for the file *{user.home}{file.separator}*krb5.keytab.
	storeKey	Set this to True to if you want the principal's key to be stored in the Subject's private credentials.

Table 10–1 (Cont.) SAML and Kerberos Login Modules Attributes and Properties

Login Module Service Name	Property	Description
	keyTab	Set this to the file name of the keytab to get principal's secret key.
	doNotPrompt	Set this to true if you do not want to be prompted for the password if credentials cannot be obtained from the cache or keytab (default is false). If set to true, authentication will fail if credentials cannot be obtained from the cache or keytab.

10.12 Configuring SAML

The SAML standard defines a common XML framework for creating, requesting, and exchanging security assertions between software entities on the Web. The SAML Token profile is part of the core set of WS-Security standards, and specifies how SAML assertions can be used for Web services security. SAML also provides a standard way to represent a security token that can be passed across the multiple steps of a business process or transaction, from browser to portal to networks of Web services.

If you use any of the following predefined policies, you must configure SAML:

- oracle/wss_saml_token_bearer_over_ssl_server_policy
- oracle/wss_saml_token_bearer_over_ssl_client_policy
- oracle/wss_saml_token_over_ssl_service_policy
- oracle/wss_saml_token_over_ssl_client_policy
- oracle/wss10_saml_token_service_policy
- oracle/wss10_saml_token_client_policy
- oracle/wss10_saml20_token_service_policy
- oracle/wss10_saml20_token_client_policy
- oracle/wss10_saml_token_with_message_protection_client_policy
- oracle/wss10_saml_token_with_message_protection_service_policy
- oracle/wss10_saml20_token_with_message_protection_client_policy
- oracle/wss10_saml20_token_with_message_protection_service_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy
- oracle/wss10_saml_hok_token_with_message_protection_service_policy
- oracle/wss10_saml_hok_token_with_message_protection_client_policy
- oracle/wss10_saml_token_with_message_integrity_service_policy
- oracle/wss10_saml_token_with_message_integrity_client_policy
- oracle/wss11_saml_token_with_message_protection_service_policy
- oracle/wss11_saml_token_with_message_protection_client_policy
- oracle/wss11_saml20_token_with_message_protection_service_policy

- oracle/wss11_saml20_token_with_message_protection_client_policy

The following sections provide more information about SAML configuration:

- "How the SAML Token is Validated" on page 10-65
- "How to Configure SAML Web Service Client at Design Time" on page 10-65
- "Including User Attributes in the Assertion" on page 10-66
- "Including User Roles in the Assertion" on page 10-67
- "How to Configure Oracle Platform Security Services (OPSS) for SAML Policies" on page 10-67
- "Adding an Additional SAML Assertion Issuer Name" on page 10-67
- "Configuring Web Service Clients for Identity Switching" on page 10-78
- "Defining a Trusted Distinguished Name (DN) List for SAML Signing Certificates" on page 10-69
- "Using Anonymous Users with SAML Policies" on page 10-69
- "Configuring Web Service Clients for Identity Switching" on page 10-78

10.12.1 How the SAML Token is Validated

The SAML login module verifies the SAML tokens on behalf of the Web service. The SAML login module then extracts the username from the verified token and (indirectly) passes it to Oracle Platform Security Services (OPSS) to complete the authentication.

10.12.1.1 Which Authentication Provider is Used?

Any configured Authentication provider as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59 can then be invoked.

10.12.2 How to Configure SAML Web Service Client at Design Time

Follow the steps described in this section to configure the SAML Web service client at design time. (If you attach the SAML policies to the Web service client at deploy time, you do not need to configure these properties and they are not exposed in Fusion Middleware Control.)

You can also include user roles in the assertion and change the SAML assertion issuer name, as described in subsequent sections.

10.12.2.1 Configure the Username for the SAML Assertion

For a JSE client application, configure the username as a BindingProvider property:

```
Map<String,Object>  reqContext = ((BindingProvider) proxy).getRequestContext()
   reqContext.put( BindingProvider.USERNAME_PROPERTY, "jdoe")
```

where *proxy* refers to the Web service proxy used for invoking the actual Web service.

For a Java EE client, if the user is already authenticated and a subject is established in the container, then the username is obtained from the subject automatically and no additional configuration is required.

For example, if user *jdoe* is already authenticated to the Java EE application and you are making a Web service call from that Java EE application, the username *jdoe* will be automatically propagated.

However, if the user is not authenticated, then you need to configure the username in the BindingProvider as in the JSE case.

10.12.3 Including User Attributes in the Assertion

SAML client policies include the `user.attributes` property that you can use to add user attributes to the SAML assertion.

To do this, you specify the attributes to be included as a comma-separated list. For example, `attrib1,attrib2`. The attribute names you specify must exactly match valid attributes in the configured identity store.

`user.attributes` requires that the Subject is available and `subject.precedence` is set to true. (If subject.precedence is true, the user name to create the SAML assertion is obtained only from the Subject.)

The Oracle WSM runtime reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion.

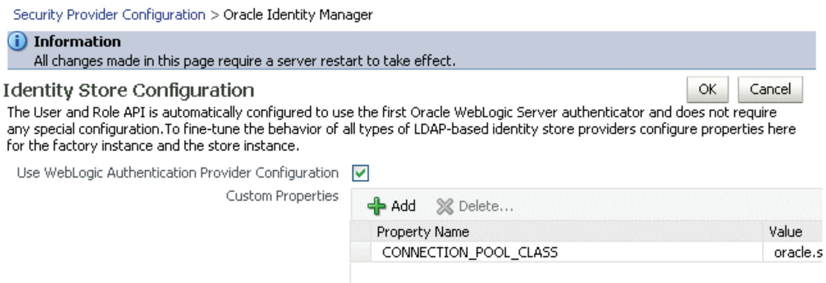
The `user.attributes` property is supported for a single identity store, and by default only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

If you have more than one identity store configured, and you want to search for the user in all identity stores, follow these steps to enable searching in all configured identity stores.

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the identity store provider. Select the domain.
2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Security**, and then **Security Provider Configuration**.
3. In the Identity Store Provider section of the page, click Configure to configure parameters that interact with the identity store.

   The Identity Store Configuration page is displayed, as shown in Figure 10–24.

Figure 10–24 Identity Store Configuration Page



Security Provider Configuration > Oracle Identity Manager
Information
Identity Store Configuration
OK
Cancel
Custom Properties
Add
Delete...

All changes made in this page require a server restart to take effect.

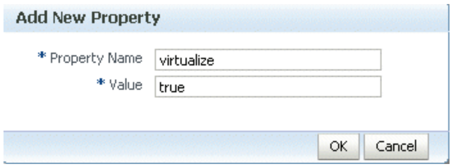
The User and Role API is automatically configured to use the first Oracle WebLogic Server authenticator and does not require any special configuration.To fine-tune the behavior of all types of LDAP-based identity store providers configure properties here for the factory instance and the store instance.

Use WebLogic Authentication Provider Configuration ☑

Property Name	Value
CONNECTION_POOL_CLASS	oracle.s

4. Click **Add** to add a custom property.
5. Add the property "virtualize" with a value of "true", as shown in Figure 10–25.

Figure 10–25 Adding the virtualize property



* Property Name: virtualize

* Value: true

6. Click **OK** to submit the changes.
7. Restart Fusion Middleware Control.

10.12.4 Including User Roles in the Assertion

You can pass the user's role as an attribute statement in the SAML assertion. To do this at post-deploy time, configure the *user.role.include* property to "true." The default value in the policy is "false."

To configure the user's role at design time, set the *user.role.include* property to "true" in the BindingProvider.

10.12.5 How to Configure Oracle Platform Security Services (OPSS) for SAML Policies

Follow these steps to configure OPSS for the predefined SAML policies:

1. Configure the SAML login module, as described in "Configuring the SAML and Kerberos Login Modules" on page 10-60.

   By default, the SAML assertion issuer name is `www.oracle.com`. The `saml.issuer.name` client property must be `www.oracle.com` if you are using the predefined SAML policies (or assertions) on both the Web service client and Web service sides. Therefore, you can generally use the defaults and not configure any issuer.

   See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for information on adding an additional issuer.

2. Configure the Authentication provider in the WebLogic Server Administration Console.
3. If you will be using policies that involve signatures related to SAML assertions (for example, SAML holder-of-key policies) where a key referenced by the assertion is used to sign the message, or sender-vouches policies where the sender's key is used to sign the message, you need to configure keys and certificates for signing and verification, as described in "Configuring Keystores for Message Protection" on page 10-9.
4. If you will be using policies that require SSL, you need to configure SSL, as described in "Configuring Keystores for SSL" on page 10-36.

10.12.6 Adding an Additional SAML Assertion Issuer Name

The SAML issuer name is generally `www.oracle.com` if you are using the predefined SAML policies (or assertions) on both the Web service client and Web service sides. Therefore, you can generally use the defaults and not configure any issuer.

There are two circumstances in which you need to add additional issuers:

- For a SAML predefined Web service policy or assertion, you set a value for the `saml.trusted.issuer` property. If you set a value for this property, you must add that trusted issuer to the Issuers list.

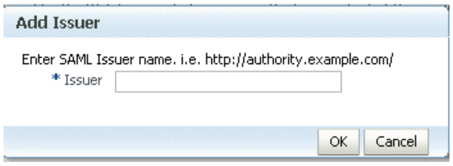
- For a SAML predefined Web service policy or assertion, you set a value for the `saml.issuer.name` property. If you set a value for this property, you must add that trusted issuer to the Issuers list with the same value.
- If a different client, for instance .NET/STS, is talking to a Web service protected by a predefined SAML policy, then you need to add that issuer to the Issuers list.

> **Note:** Although you can add an issuer by the mechanism described here, it is for backward compatibility only. The preferred method is to configure trusted issuers at the platform policy configuration level. See "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23.
>
> There is a hierarchy that determines how trusted issuers are determined:
>
> 1. First, the list of trusted issuers configured for policies (or overridden) is checked and used.
> 2. If not configured for policies (or overridden), the configuration at the platform policy configuration is checked and used. See "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23.
> 3. If not configured for policies (or overridden) or configured at the platform policy configuration, only then is the Issuers list defined in the SAML login module used.
>
> If you do define the SAML issuers by the mechanism described here, the issuers are persisted in `jps-config.xml`, and any changes take effect only after a restart of the domain.

To add an additional SAML assertion issuer to the Issuers list:

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to add the issuer. Select the domain.
2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Security**, and then **Security Provider Configuration**.
3. Select the SAML or SAML2 login module as appropriate and click **Edit**.
4. From the SAML Specific Attributes section of the page, click **Add** to add an additional issuer name, as shown in Figure 10–26.

Figure 10–26 Adding a SAML Issuer to the Login Module


Add Issuer

Enter SAML Issuer name. i.e. http://authority.example.com/

* Issuer

5. For a client policy, at deploy time, specify a value for `saml.issuer.name` on the **Configurations** page for the SAML client policy, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The default value in the policy is `www.oracle.com`.

   To configure the issuer at design time, set the `saml.issuer.name` property in the BindingProvider.

10.12.7 Defining a Trusted Distinguished Name (DN) List for SAML Signing Certificates

For additional security, you can define a list of trusted DNs for SAML signing certificates.

By default, Oracle WSM checks the incoming issuer name against the list of configured issuers, and checks the SAML signature against the configured certificates in the Oracle WSM keystore. If you define a trusted DNs list, Oracle WSM also verifies that the SAML signature is signed by the particular certificate(s) that is associated with that issuer.

Configuration of the trusted DNs list is optional; it is available for users that require more fine-grained control to associate each issuer with a list of one or more signing certificates. If you do not define a list of DNs for a trusted issuer, then Oracle WSM allows signing by any certificate, as long as that certificate is trusted by the certificates present in the Oracle WSM keystore.

For more information about defining a trusted DNs list for SAML signing certificates, see "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23.

10.12.8 Using Anonymous Users with SAML Policies

All SAML policies allow anonymous users to be propagated. For example if you have an ADF application that can work with either authenticated users or an anonymous user, and this ADF application needs to make a call to a Web service and propagate the current user, then you can propagate both the authenticated users and the anonymous user using any of the SAML policies. From the security perspective, propagating the anonymous user over SAML is equivalent to the client not sending any authentication tokens to the service.

Allowing anonymous users over SAML is provided as a convenience so that you can have one policy that supports both authenticated and anonymous users. Note, however, that anonymous propagation over SAML is non-standard and will not interoperate with other vendors. It should only be used when both the client and the Web service are using Oracle WSM.

10.13 Using JSON Web Token (JWT) with Oracle WSM

JSON Web Token (JWT) is a means of representing claims to be transferred between two parties. JWT is a compact token format intended for space- constrained environments such as HTTP Authorization headers and URI query parameters.

The following sections provide a high-level example and the necessary configuration steps to use JWT with Oracle WSM:

- Example JWT Use Case (High Level Steps)
- Configuring the Client and Service for Propagating the JWT Token

10.13.1 Example JWT Use Case (High Level Steps)

This example use case demonstrates the following scenario:

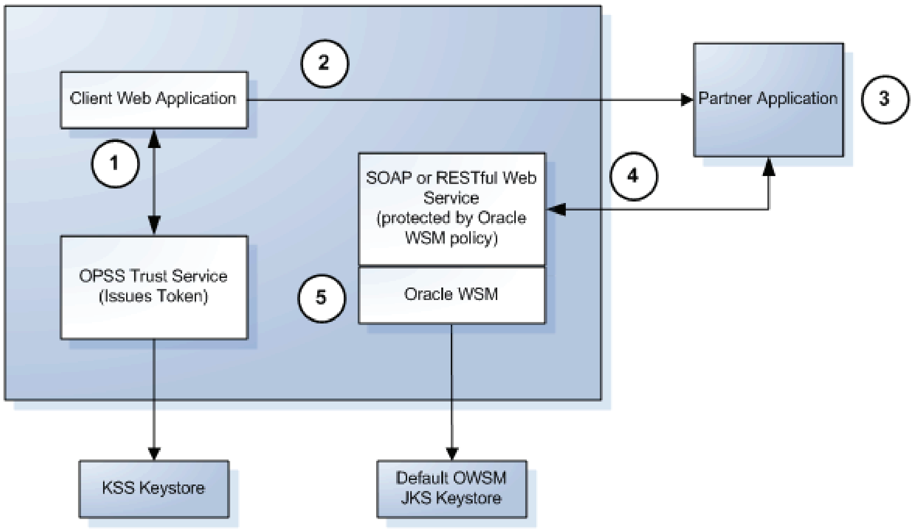
- Using a browser, an end user points to a partner-embedded user interface within a client Web application.
- To display user-customized information and perform further transactional work, the browser calls an external partner application and authenticates/authorizes.

- The partner application retrieves end user related data (such as email) from the client application.
- The transactional work is completed by calling back to a secure SOAP or RESTful Web service (on behalf of the end user that initially logged into the client Web application, thus performing identity propagation).

The following high-level steps describe how to implement this example use case by passing a JWT token between the client Web application, the partner application, and an Oracle WSM protected Web service.

Figure 10–27 illustrates this example.

Figure 10–27 Propagating JWT Token Example


Client Web Application
2
Partner Application
3
1
SOAP or RESTful Web Service (protected by Oracle WSM policy)
4
OPSS Trust Service (Issues Token)
5
Oracle WSM
KSS Keystore
Default OWSM JKS Keystore

Note: This example assumes the client and service have been properly configured as described in "Configuring the Client and Service for Propagating the JWT Token" on page 10-72.

1. The client Web application invokes the OPSS Trust Service to obtain the JWT token.

   To do so, it needs to pass the UserID to the TokenManager. Optionally, it can also pass other UserInfo (such as email, firstName, and so on) and AppInfo (such as orgID, opportunityID) as new Claims, as shown in Example 10–8.

Example 10–8 Invoking OPSS Trust Service to Obtain JWT Token

```
javax.security.auth.Subject subject =
javax.security.auth.Subject.getSubject(AccessController.getContext());
 String userName = oracle.security.jps.util.SubjectUtil.getUserName(subject);
  JpsContextFactory ctxFactory = JpsContextFactory.getContextFactory();
        JpsContext jctx = ctxFactory.getContext();
        TrustService trustService = jctx.getServiceInstance(TrustService.class);
        final TokenManager tokenMgr = trustService.getTokenManager();

        // Issue token
    final ExtendedTokenContext ectx = (ExtendedTokenContext)
tokenMgr.createTokenContext(TokenConfiguration.PROTOCOL_EMBEDDED);
        ectx.setTokenType(JwtToken.JWT);
        TrustToken tsToken = new TrustToken(TokenConstants.TOKEN_TYPE_USERID,
userName);
        ectx.setTrustToken(tsToken);
        ectx.setIssuer("www.oracle.com");

        Map<String, Object> ctxProperties = ectx.getOtherProperties();
        ctxProperties.put("trust.tokenSigningMethod", JwtToken.SIGN_
ALGORITHM.RS256.toString());
// Passing parameters like email, opportunityID
ArrayList<Object> emailValue = new ArrayList<Object>();
emailValue.add("joe.abc@example.com");
Claim emailClaim = new Claim("oracle:apps:attributes:email", emailValue);

ArrayList<Object> oppIDValue = new ArrayList<Object>();
oppIDValue.add("200");
Claim oppIDClaim = new Claim("oracle:apps:attributes:opportunityID", oppIDValue);
List<Claim> list = new ArrayList<Claim>();
list.add(emailClaim);
list.add(oppIDClaim);

ectx.setClaims(list)

        Object tokenData = AccessController.doPrivileged(new
PrivilegedExceptionAction<Object>() {
@Override
public Object run() throws Exception {
try {
return  tokenMgr.issueTrustToken(ectx);

} catch (Exception ex) {
                      throw ex;
}
}
});
           Object tokValue = ectx.getTrustToken().getTokenValue(); //base 64
encoded JWT token
```

For more information about obtaining a token from the OPSS Trust Service, see the following topics in *Oracle Fusion Middleware Application Security Guide*:

- The OPSS Trust Service
- Propagating Identities with the OPSS Trust Service

2. The client Web application passes the JWT token to the partner application in the URL parameter:

```
?token=<tokValue>
```

> **Note:** The transport channel used to pass the token must be secured using SSL.

3. The partner application extracts the token from the URL parameter and verifies the token signature.

4. The partner application passes the token extracted from the URL parameter to a SOAP or RESTful Web service protected with an Oracle WSM service policy in the Authorization:Bearer HTTP header:

```
Authorization:Bearer <tokValue>
```

Note: The transport channel used to pass the token must be secured using SSL.

---

5. Oracle WSM verifies the JWT token and establishes the subject with the user identity. Note that the user identity being propagated must exist in the identity store on the service side.

10.13.2 Configuring the Client and Service for Propagating the JWT Token

It is necessary to perform configuration on both the client side and the service side, as described in the following sections:

- Client-side Configuration for JWT Token Propagation
- Service-side Configuration for JWT Token Propagation

There is no configuration required for the customer application.

10.13.2.1 Client-side Configuration for JWT Token Propagation

The OPSS Trust Service uses the KSS keystore, and Oracle WSM uses the JKS keystore (`default-keystore.jks`). It is necessary to sync these two keystores by copying the JKS keystore into the OPSS KSS keystore, and then grant the client application the appropriate permission to access the OPSS Trust Service.

Step 1: Sync the KSS and JKS Keystores

1. Make a copy of the Oracle WSM JKS keystore:

   a. Create a directory called `kss` on your filesystem.

   b. Copy the Oracle WSM keystore `default-keystore.jks` from the `$DOMAIN_HOME/config/fmwconfig` directory into `kss` directory:

   ```
   cp default-keystore.jks default-keystore-copy.jks
   chmod 777 default-keystore-copy.jks
   ```

2. Change the `orakey` alias to `trustservice` alias in the keystore:

   ```
   keytool -changealias -alias orakey -keypass orakey -destalias trustservice
   -storepass password -keystore ./default-keystore-copy.jks
   ```

   Note: In this example, the `keypass` is `orakey` and the `storepass` is *`password`*. Be sure to use the appropriate passwords for your keystore.

   ---

3. Upload the keystore in KSS by executing the following CLI commands:

   Note: The `stripeName` and `KeystoreName` are configured as follows:

   `stripeName=opss` and `keystoreName=trustservice_ks`, which contains `KeyPair` with the alias specified as `trustservice`

   `stripeName=opss`, `keystoreName=trustservice_ts`, which contains `Certificate`, with alias specified as `trustservice`

```
svc = getOpssService(name='KeyStoreService')
svc.deleteKeyStoreEntry(appStripe='opss', name='trustservice_ks', password='',
alias='trustservice', keypassword='password')
svc.deleteKeyStoreEntry(appStripe='opss', name='trustservice_ts', password='',
alias='trustservice', keypassword='password')

svc.importKeyStore(appStripe='opss', name='trustservice_ks', password='',
aliases='trustservice', keypasswords='password', type='JKS', permission=true,
filepath=Oracle WSMkeystorecopy)
svc.importKeyStore(appStripe='opss', name='trustservice_ts', password='',
aliases='trustservice', keypasswords='password', type='JKS', permission=true,
filepath=Oracle WSMkeystorecopy)

print '--------\--list keystore
alias-\------------------------------------------'

svc.listKeyStoreAliases(appStripe='opss', name='trustservice_ks', password='',
type='*')
svc.listKeyStoreAliases(appStripe='opss', name='trustservice_ts', password='',
type='*')
```

Step 2: Grant OPSS Trust Service Permission to Client Web Application

Before it can invoke the OPSS Trust Service to obtain the JWT token, the client Web application must be granted the `TrustServiceAccessPermission` using the `grantPermission` WLST command, as follows:

```
grantPermission(codeBaseURL="file:${oracle.deployed.app.dir}/MyApp${oracle.deploye
d.app.ext}",
permClass="oracle.security.jps.service.trust.TrustServiceAccessPermission",permTar
get="appId=*", permActions="issue")
```

In this command line, replace *`MyApp`* with the name of the client application.

10.13.2.2 Service-side Configuration for JWT Token Propagation

On the service side, you must secure the Web service with an Oracle WSM policy that supports the JWT token, and configure the JWT trusted issuer for the domain.

Step 1: Secure the Web Service

Secure the SOAP Web service or RESTful servlet application using one of the following policies:

- For a SOAP Web service, use `wss11_saml_or_username_token_with_message_protection_service_policy`. For details about attaching policies to SOAP Web services, see "Attaching Policies to Web Services" on page 8-3.
- For RESTful servlet applications, use `multi_token_rest_service_policy`. For details about attaching policies to RESTful servlet applications, see "Attaching Policies to Servlet Applications" on page 8-18.

For instructions on configuring Web services with policies that enable identity switching for JWT tokens, see "Configuring Web Service Clients for Identity Switching" on page 10-78.

Step 2: Define the Trusted Issuer for the JWT Token

You define the JWT Trusted Issuer using the `setWSMTokenIssuerTrust` WLST command as described in "Defining Trusted Issuers and Managing DN Lists Using WLST" on page 14-25. For example:

```
setWSMTokenIssuerTrust('dns.jwt','www.oracle.com',['CN=weblogic, OU=Orakey Test
Encryption Purposes Only, O=Oracle, C=US'])
```

10.14 Using OAuth2 with Oracle WSM

This section describes using the Oracle Mobile and Social OAuth2 authorization framework with Oracle WSM.

The section assumes that you are familiar with both the terminology and the conceptual and configuration information described in the following sections of *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*:

- "Understanding OAuth Services"
- "Configuring OAuth Services"

10.14.1 Understanding OAuth2 with Oracle WSM

OAuth2 support in Oracle WSM is based on "The OAuth 2.0 Authorization Framework" specification, which is available at `http://tools.ietf.org/html/rfc6749`. Oracle WSM uses the Oracle Mobile and Social OAuth2 service as the authorization server for the OAuth2.0 protocol interactions.

Oracle WSM allows web service clients to interact with the Mobile and Social OAuth2 server implementation for both SOAP and REST web services, for "2-legged" authorization.

10.14.1.1 2-Legged Authorization

In 2-legged OAuth2, the interaction is application-to-application without user consent.

The client requests authorization from the resource owner. In response, the client receives an **authorization grant**, which is a credential representing the resource owner's authorization. Then:

1. The client requests an access token (AT) by authenticating with the authorization server and presenting the authorization grant.
2. The authorization server authenticates the client and validates the authorization grant, and if valid, issues an AT.
3. The client requests the protected resource from the resource server and authenticates by presenting the AT.
4. The Oracle WSM server side agent validates the AT and accepts the request if valid or rejects the request if invalid.

You attach the OAuth2 client policy such as `oracle/http_oauth2_token_client_policy` and the `oracle/oauth2_config_client_policy` to the client application. The required `token.uri` property of the `oracle/oauth2_config_client_policy` policy specifies the OAuth2 server token endpoint.

You also attach any of the following Oracle WSM JWT service policies to the web service. The Oracle WSM server-side agent validates the access token.

- `oracle/http_jwt_token_service_policy`
- `oracle/http_jwt_token_over_ssl_service_policy`
- `multi_token_rest_service_policy`
- `multi_token_over_ssl_rest_service_policy`

(See "Using JSON Web Token (JWT) with Oracle WSM" on page 10-69 for information on the Oracle WSM JWT service policies.)

Supported Authorization Grant Types in 2-Legged Authorization

As previously described, an **authorization grant** is a credential representing the resource owner's authorization.   Oracle WSM supports the following authorization grant types in 2-legged authorization. You specify which types you want to use when you configure the OAuth2 OWSM client profile, as described in "Prerequisites for Using the Oracle WSM OAuth2 Policies" on page 10-76.

- Client credentials grant - In this case the client credentials are sent in the "Authorization: Basic" HTTP header as explained in Client Credentials Grant - OAuth2.0 Authorization Framework.

  You set the client token policy `oauth2.client.csf.key` property to specify the user name and password to use. Ensure that the `federated.client.token` property is set to false.

- Client credentials JWT (Federation use case) - In this case the client credentials are sent in the form of a JWT assertion, as explained in Using JWTs for Client Authentication.

  Oracle WSM generates the JWT token locally based on client credentials stored in the Oracle WSM credential store. You set the client token policy `federated.client.token` property to specify whether a JWT token is generated for the client using the values of the `oauth2.client.csf.key` and `keystore.sig.csf.key` properties.

- Client credentials are sent in the Basic Auth Header, plus user credentials in the JWT assertion, as explained in Client Credentials Grant - OAuth2.0 Authorization Framework and Using JWTs for Client Authentication.

  You set the client token policy `oauth2.client.csf.key` property to specify the user name and password to use in the Basic Auth Header.

- Client credentials are sent in the JWT assertion, plus user credentials in the JWT assertion, as explained in Using JWTs for Client Authentication.

How Client Credentials Are Determined in 2-Legged Authorization

The client credential is always included in the request to the OAuth2 server. The `federated.client.token` property determines whether the JWT is used for the client ID in the client credential, or whether the client ID and password are used for the client credential.

- If `federated.client.token` is true (the default), then the JWT is used for the client ID in the client credential.
- If `federated.client.token` is false, then the client ID and password are used for the client credential.

Relationship of User Credentials, Client Credentials, and Subject in 2-Legged Authorization

The `subject.precedence` property specifies the location from which the subject used to create the JWT token is obtained.

As shown in Table 10–2, if `subject.precedence` is set to true, the user name to create the JWT token is obtained only from the authenticated subject.

If `subject.precedence` is set to false, the user name to create the JWT token is obtained only from the `csf-key` property.

Table 10–2 User Credential, Subject, and Access Token

subject.precedence	csf-key	Authenticated User Subject	Client Credential	User Credential	Access Token Principal/Subject
True (default)	N/A	Available	See How Client Credentials Are Determined in 2-Legged Authorization.	JWT for authenticated end user.	End-user name.
True (default)	N/A	Not available	See How Client Credentials Are Determined in 2-Legged Authorization.	Not included	Client ID
False	Not configured (default)	N/A	See How Client Credentials Are Determined in 2-Legged Authorization.	Not included	N/A
False	Configured	N/A	See How Client Credentials Are Determined in 2-Legged Authorization.	JWT for the identity from the csf-key entry.	The user name from the csf-key/user name is configured.

10.14.2 Prerequisites for Using the Oracle WSM OAuth2 Policies

Perform these steps to configure Mobile and Social OAuth2 for use with Oracle WSM. The steps assume that you are familiar with the configuration information described in *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

1. Enable OAuth services, as described in "Enabling OAuth Services".
2. For REST, create an `Authorization and Consent Service Plug-ins` plug-in profile, as described in Configuring OAuth Plug-Ins.

   Consider the following example:

   - Coarse grain (Recommended)
     - Name: `PermissionRESTPlugin`
     - Description: `Permission Authorization REST  Coherence Plug-in`
     - Security Handler Class: *`oracle.security.idaas.oauth.consent.impl.CoherenceAuthorizationUserConsentImpl`*
   - Fine Grained (with endpoint)
     - Name: `PermissionRESTPlugin`
     - Description: Permission Authorization REST Callback Plug-in
     - Security Handler Class: `oracle.security.idaas.oauth.consent.impl.AuthorizationRESTCallbackImpl`

   - Attributes: `rest.permission.service.endpoint=`*`endpoint`*.

3. Create a resource server profile, as described in "Creating an OAuth Resource Server".

   If you created an `Authorization and Consent Service Plug-ins` in Step2, specify it in the resource server profile.

4. Configure the client profile, as described in "Understanding OAuth Client Profiles Configuration". Use the **Create Oauth Web Client** page.

   a. Specify a client ID and a client secret. Make a note of these values, you must use them when you later configure the OWSM client policy.

   b. Under Privileges, select **Allow Access to All Scopes**.

   c. Under Privileges, select the grant types you want to use:

      For 2-legged authorization, select **Client Credentials** and **JWT bearer**.

5. Update the OAuth2 server profile configuration, as described in "Editing or Deleting an OAuth Service Profile".

   If you are using JWT Bearer as one of the client credentials:

   - Set the `jwt.issuer` to www.oracle.com
   - Set `jwt.CryptoScheme` to RS256
   - Set `jwt.trusted.issuer.1` to http://www.oracle.com
   - Set `jwt.trusted.issuer.size` to 1.

6. Create an `oauth2.client.csf.key` key of type password in the Oracle WSM credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19.

   Use the client ID from Step 4 as the username and the client secret as the password.

   > **Note:** The username and password must represent a valid user in the OPSS identity store.

7. Export the Oracle WSM client certificate and import it into the OAuth2 server truststore to verify the JWT token generated by the Oracle WSM client.

   See "Configuring Keystores for Message Protection" on page 10-9 for information on exporting the Oracle WSM client certificate.

8. Export the OAuth2 server certificate and import it into the Oracle WSM trust store on the web service side so that Oracle WSM can verify the signed AT generated by the OAuth2 server.

   The default OAuth Service Profile included in the default domain uses the Java Keystore (JKS) included with Oracle Access Management. See "Configuring OAuth Services" for information on the OAuth2 keystore.

   See "Configuring Keystores for Message Protection" on page 10-9 for information on importing the OAuth2 server certificate into the Oracle WSM keystore.

Proceed to Chapter 11, "Configuring Policies" for information on configuring the Oracle WSM OAuth2 policies.

See Chapter 8, "Attaching Policies to Web Services" for information on attaching policies.

10.15 Configuring Web Service Clients for Identity Switching

Oracle WSM includes SAML and JWT client policies that enable identity switching. Identity switching means that the policy propagates a different identity than the one based on the authenticated Subject. The identity switch policies are:

- `wss11_saml_token_identity_switch_with_message_protection_client_policy` – This policy can be used for outbound SOAP requests using mechanisms described in WS-Security 1.1. For more information, see "oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy" on page 11-90.
- `wss_saml_token_bearer_identity_switch_client_policy` – This policy can be attached to any SOAP client endpoint. For more information, see "oracle/wss_saml_token_bearer_identity_switch_client_policy" on page 11-57.
- `http_jwt_token_identity_switch_client_policy` – This policy can be enforced on any HTTP-based, SOAP, or REST client endpoint. For more information, see "oracle/http_jwt_token_identity_switch_client_policy" on page 11-8.

You might have a scenario in which your SOA or REST application needs to specify which user identity to use in client-side Web service policies, and then dynamically switch the user associated with the SAML or JWT token in the outbound Web service request. Instead of using the username from the Subject, these policies allow you to set a new user name when sending the SAML or JWT Web service request.

For example, the `wss11_saml_token_identity_switch_with_message_protection_client_policy` policy creates the SAML token based on the user ID set via the property `javax.xml.ws.security.auth.username`.

Compatibility

The identity switch client policies are compatible with the following policies on the Web service:

- `wss11_saml_token_identity_switch_with_message_protection_client_policy`
  - `wss11_saml_token_with_message_protection_service_policy`
  - `wss11_saml_or_username_token_with_message_protection_service_policy`
- `wss_saml_token_bearer_identity_switch_client_policy`
  - `wss_saml_token_bearer_service_policy`
  - `wss_saml_bearer_or_username_token_service_policy`
- `http_jwt_token_identity_switch_client_policy`
  - `http_jwt_token_service_policy`
  - `multi_token_rest_service_policy`
  - `wss11_saml_or_username_token_with_message_protection_service_policy`

Use Case Example

Consider the following use case in which a Web service client calls a SOA application, which in turn becomes the client for a Web service.

```
client -> SOA -> web service
```

In this use case:

- The client is secured with the wss11_username_with_message_protection_client_policy or with a similar client policy over SSL. It communicates with the SOA entry point as user end_user1.
- The SOA entry point is protected by a corresponding type of service policy, such as wss11_username_with_message_protection_service_policy or wss_username_over_ssl_service_policy. The SOA application authenticates the end user and establishes the Subject based on end_user1. However, it wants to propagate a different identity to the external Web service.

  Therefore, to do identity switching, attach an appropriate identity switch policy, such as the wss11_saml_identity_switch_message_protection_client_policy policy for SAML tokens, to the SOA reference binding component.
- The username that is propagated is determined dynamically by the BPEL process, which is a component in the SOA application. The username is set as BPEL property javax.xml.ws.security.auth.username with the dynamically determined username value, as described in "Directly from the Message Context" on page 10-79. The external Web service can be protected by wss11_saml_with_message_protection_service_policy. It receives the switched user and not end_user1.
- A similar scenario can be used by a Java EE application (replacing SOA in this scenario with the Java EE application) that establishes the Subject based on an end user but then needs to propagate a different identity. In the case of Java EE, you can set the user name programmatically as follows:

  ```
  ((BindingProvider) port).getRequestContext().put(BindingProvider.USERNAME_
  PROPERTY, config.get(USERNAME));
  ```
- Use Fusion Middleware Control or WLST to add the WSIdentityPermission permission to the SOA reference binding component, as described in "Set the WSIdentityPermission Permission" on page 10-80.

  The identity switch policies require that an application to which the policy is attached must have the WSIdentityPermission permission. That is, applications from which Oracle WSM accepts the externally-supplied identity must have the WSIdentityPermission permission.

  This is to avoid potentially rogue applications from providing an identity to Oracle WSM.

Note: The identity switch policies (wss11_saml_token_identity_switch_with_message_protection_client_policy, wss_saml_token_bearer_identity_switch_client_policy, and http_jwt_token_identity_switch_client_policy) disable local optimization (see "Configuring Local Optimization for a Policy" on page 11-142 for SOA-to-SOA interactions on the same server.)

10.15.1 How the Username Is Picked Up by an Identity Switch Policy on the Client Side

This section describes the ways in which the username can be picked up for use by an identity switch policy on the client side.

10.15.1.1 Directly from the Message Context

For SOA:

The SOA composite has a BPEL process as one SOA service component. A BPEL process provides process orchestration and storage of synchronous and asynchronous processes. Therefore, you can define a BPEL property with the exact name `javax.xml.ws.security.auth.username`. The value for this property can be the identity that the SOA application wants to propagate, which could potentially be determined dynamically by the BPEL process.

For Java EE:

Set the `BindingProvider.USERNAME_PROPERTY` property.

10.15.1.2 From the csf-key in the Message Context

Set the `SecurityConstants.ClientConstants.WSS_CSF_KEY` property in the request context.

10.15.1.3 Overriding the csf-key in the Client Policy

Provide a value for the `ClientConstants.WSS_CSF_KEY` field for any given identity switch policy. For example, for an HTTP identity switch policy, the csf-key configuration override can be configured by client, as follows:

```
PropertyFeature csfKey = new PropertyFeature(
SecurityConstants.ClientConstants.CO_CSF_KEY, "Oracle WSMtest.credentials");
```

10.15.2 Set the WSIdentityPermission Permission

The Web service client (for example, the SOA reference binding component) to which you attached the `wss11_saml_token_identity_switch_with_message_protection_client_policy` policy must have the `oracle.wsm.security.WSIdentityPermission` permission.

10.15.2.1 Using Fusion Middleware Control

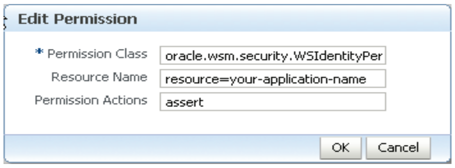
To use Fusion Middleware Control to add the `oracle.wsm.security.WSIdentityPermission` permission to the SOA reference binding component as a System Grant, perform these steps:

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the application. Select the domain.
2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Security**, and then **System Policies**. System policies are the system-wide policies applied to all applications deployed to the current WebLogic Domain.
3. From the **System Policies** page, select the arrow icon in the **Permission** field to search the system security grants.
4. Select one of the codebase permissions to use as a starting point and click **Create Like**.
5. In the **Grant Details** section of the page, enter `file:${common.components.home}/modules/oracle.wsm.agent.common_${jrf.version}/wsm-agent-core.jar` in the **Codebase** field.

> **Note:** When defining the grant details, Oracle recommends that you avoid using product version numbers in the directory or JAR names. This will minimize impact when upgrading to a new release in the future.

6. In the **Permissions** section of the page, select the starting point permission class and click **Edit**.
7. Enter `oracle.wsm.security.WSIdentityPermission` in the **Permission Class** field. The resource name is the composite name for SOA, and the application name for a Java EE client. The action is always *assert*, as shown in Figure 10–28.

Figure 10–28 Editing the WSIdentityPermission



Edit Permission

* Permission Class: oracle.wsm.security.WSIdentityPer

Resource Name: resource=your-application-name

Permission Actions: assert

10.15.2.2 Using WLST

To use WLST to add the `oracle.wsm.security.WSIdentityPermission` permission, execute the following command:

```
grantPermission(codeBaseURL="file:${common.components.home}/modules/
oracle.wsm.agent.common_${jrf.version}/wsm-agent-core.jar",
     permClass="oracle.wsm.security.WSIdentityPermission",
     permTarget="resource=yourAppName",
     permActions="assert")
```

In this command:

- `codeBaseURL` must point to `wsm-agent-core.jar`.
- `permTarget` syntax is `"resource=yourAppName/compositeName"`. The resource name is the composite name for SOA, and the application name for a Java EE client.
- `permActions` is always `"assert"`.

10.16 Propagating Identity Context with Oracle WSM

Identity Context allows organizations to meet growing security threats by using the context-aware policy management and authorization capabilities built into the Oracle Access Management platform. Identity Context secures access to resources using traditional security controls (such as roles and groups) and as dynamic data established during authentication and authorization (such as authentication strength, risk levels, device trust, and so on).

For example, an application could use Identity Context to:

- Disable a particular business function if the user is not authenticated using a strong credential such as smart card.
- Secure access to a transaction based on the identity data supplied by a business partner (by using Identity Federation) with whom the organization does business.
- Request additional authentication credentials if it detects that access is originating from a location known for fraudulent activities.
- Limit the scope of administrative authority if the Administrator's industry certification (as maintained by a third party) has expired.
- Disable certain business functions if it detects that access is originating from an unknown device.

Oracle WSM can propagate the Identity Context from the Web service client to the Web service, and then make it available ("publish it") to other components for authentication and authorization purposes.

The Identity Context is opaque to your Web service client and Web service, and you need not perform any additional coding or processing in your Web service client or Web service to support it once you enable Identity Context propagation for your policies.

Note: Identity Context propagation is not supported for SOA, WebCenter, and WebLogic (Java EE) Web service applications.

For more information on Identity Context, configuring the Identity Context Service, and using the Identity Context API, see "Using Identity Context" in *Administrator's Guide for Oracle Access Management*.

10.16.1 Using SAML and JWT Policies to Propagate Identity Context

To use this feature, you must specifically enable Identity Context propagation by using the `propagate.identity.context` configuration property for both the Web service policy and the Web service client policy. That is, Oracle WSM can propagate the Identity Context only if you specifically allow both the Web service client policy and Web service policy to do so.

Oracle WSM propagates the Identity Context from Web service clients to Web services by using SAML 1.1, SAML 2.0, or JWT assertions. Therefore, only SAML and JWT policies include the `propagate.identity.context` configuration property.

The following Oracle WSM policies contain the `propagate.identity.context` configuration property:

- oracle/http_jwt_token_service_policy and oracle/http_jwt_token_client_policy
- oracle/http_jwt_token_over_ssl_service_policy and oracle/http_jwt_token_over_ssl_client_policy
- oracle/http_saml20_token_bearer_service_policy and oracle/http_saml20_token_bearer_client_policy
- oracle/http_saml20_token_bearer_over_ssl_service_policy and oracle/http_saml20_token_bearer_over_ssl_client_policy
- oracle/wss_saml_or_username_token_service_policy
- oracle/wss_saml_or_username_token_over_ssl_service_policy
- oracle/wss_saml_token_bearer_over_ssl_service_policy and oracle/wss_saml_token_bearer_over_ssl_client_policy
- oracle/wss_saml_token_over_ssl_service_policy and oracle/wss_saml_token_over_ssl_client_policy
- oracle/wss_saml20_token_bearer_over_ssl_service_policy and oracle/wss_saml20_token_bearer_over_ssl_client_policy
- oracle/wss_saml20_token_over_ssl_service_policy and oracle/wss_saml20_token_over_ssl_client_policy
- oracle/wss10_saml_token_service_policy and oracle/wss10_saml_token_client_policy

- oracle/wss10_saml_token_with_message_integrity_service_policy and oracle/wss10_saml_token_with_message_integrity_client_policy
- oracle/wss10_saml_token_with_message_protection_service_policy and oracle/wss10_saml_token_with_message_protection_client_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy and oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy
- oracle/wss10_saml20_token_service_policy and oracle/wss10_saml20_token_client_policy
- oracle/wss10_saml20_token_with_message_protection_service_policy and oracle/wss10_saml20_token_with_message_protection_client_policy
- oracle/wss11_saml_token_with_message_protection_service_policy and oracle/wss11_saml_token_with_message_protection_client_policy
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy
- oracle/wss11_saml20_token_with_message_protection_service_policy and oracle/wss11_saml20_token_with_message_protection_client_policy

10.16.2 Configuring Identity Context Propagation: Main Steps

You can specify a value for `propagate.identity.context` on the **Configurations** tab in the policy as described in this section, or override it when you attach the policy.

Note: For JWT token policies, the **Configurations** tab is not available. For these policies, you must edit the properties in the Assertion Content XML directly, as described in "Configuring Identity Context Propagation for JWT Token Policies" on page 10-84.

For information about overriding the `propagate.identity.context` property after you attach the policy, see the following topics:

- "Attaching Web Service Policies Permitting Overrides" on page 8-25
- "Attaching Client Policies Permitting Overrides" on page 8-31
- "Overriding Configuration Properties for Globally Attached Policies" on page 9-22

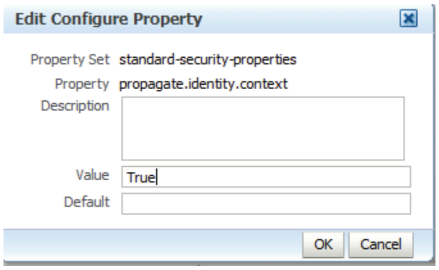
Note: Oracle recommends that you do not edit the predefined policies so that you will always have a known set of valid policies. You can, however, create new policies (both server and client) using the predefined policies listed in "Using SAML and JWT Policies to Propagate Identity Context" on page 10-82. For additional information about creating a new policy, see "Creating a Web Service Policy from an Existing Policy" on page 7-6. Once you have created the new policy, you can edit the policy and set the `propagate.identity.context` property as described below.

By default, the `propagate.identity.context` configuration property is not set, which is equivalent to `False`. To use Identity Context propagation, you must specifically set `propagate.identity.context` to `True`.

10.16.2.1 Configuring Identity Context Propagation for SAML Policies

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure Identity Context propagation. Select the domain.
2. In the content pane, select **WebLogic Domain**, then **Web Services**, and then **Policies**.
3. Select the SAML policy for which you want to enable Identity Context propagation and click **Edit**. Remember that you have to enable Identity Context propagation for both the Web service client and Web service policies.
4. On the Edit Policy page, select the **Configurations** tab.
5. Select `propagate.identity.context` from the list of properties and click **Edit**.
6. Change the value field to `True` and click **OK**, as shown in Figure 10–29.

Figure 10–29 Setting the propagate.identity.context Property to True



Edit Configure Property

Property Set standard-security-properties

Property propagate.identity.context

Description

Value True

Default

7. Click **Save** to submit the changes.
8. Repeat Steps 3 through 7 for the corresponding client or service policy, as appropriate.

10.16.2.2 Configuring Identity Context Propagation for JWT Token Policies

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure Identity Context propagation. Select the domain.
2. In the content pane, select **WebLogic Domain**, then **Web Services**, and then **Policies**.
3. Select the JWT policy for which you want to enable Identity Context propagation and click **Edit**. Remember that you have to enable Identity Context propagation for both the Web service client and Web service policies.
4. On the Edit Policy page, locate the `propagate.identity.context` property in the Assertion Content section of the page.
5. Change the value field to `true` as follows:

```
<orawsp:Property orawsp:contentType="optional"
orawsp:name="propagate.identity.context" orawsp:type="string">
<orawsp:Value>true</orawsp:Value>
</orawsp:Property>
```

6. Click **Save** to submit the changes.
7. Repeat Steps 3 through 6 for the corresponding client or service policy, as appropriate.

10.17 Using Kerberos Tokens

Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9) provides support for Kerberos tokens with the following predefined policies:

- oracle/wss11_kerberos_token_client_policy
- oracle/wss11_kerberos_token_service_policy
- oracle/wss11_kerberos_token_with_message_protection_client_policy
- oracle/wss11_kerberos_token_with_message_protection_service_policy
- oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy
- oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy

You may also create a policy using the following assertion templates:

- oracle/http_spnego_token_client_template
- oracle/http_spnego_token_service_template
- oracle/wss11_kerberos_token_client_template
- oracle/wss11_kerberos_token_service_template
- oracle/wss11_kerberos_token_with_message_protection_client_template
- oracle/wss11_kerberos_token_with_message_protection_service_template

See Appendix B, "Predefined Policies" and Appendix C, "Predefined Assertion Templates" for more information on these policies and assertions.

Follow the steps described in this section to configure Kerberos for use by the Web service client and Web service.

You can also use Microsoft Active Directory with the Key Distribution Center (KDC). See "Using Active Directory with Kerberos and Message Protection" on page 10-90.

- "Initializing and Starting the MIT Kerberos KDC" on page 10-85
- "Creating Principals" on page 10-86
- "Configuring the Web Service Client to Use the Correct KDC" on page 10-86
- "Setting the Service Principal Name In the Web Service Client" on page 10-87
- "Setting the Service Principal Name In the Web Service Client at Design Time" on page 10-87
- "Configuring the Web Service to Use the Correct KDC" on page 10-88
- "Using the Correct Keytab File in Enterprise Manager" on page 10-88
- "Authenticating the User Corresponding to the Service Principal" on page 10-88
- "Creating a Ticket Cache for the Web Service Client" on page 10-89
- "Configuring Kerberos With SPNEGO Negotiation" on page 10-89

10.17.1 Initializing and Starting the MIT Kerberos KDC

Initialize the Key Distribution Center (KDC) database. For example, on UNIX you might run the following command as root, where *example.com* is your default realm:

```
root# /usr/kerberos/sbin/krb5_util -r example.com -s
```

Start the kerberos service processes.   For example, on UNIX you might run the following commands as root.:

```
root# /usr/kerberos/sbin/krb5kdc &
root# /usr/kerberos/sbin/kadmind &
```

10.17.2 Creating Principals

Create two accounts in the KDC user registry. The first account is for the end user; that is, the Web service client principal. The second account is for the Web service principal.

One way to create these accounts is with the kadmin.local tool, which is typically provided with MIT KDC distributions. For example:

```
>sudo su - # become root
>cd /usr/kerberos/sbin/kadmin.local
>kadmin.local>addprinc fmwadmin -pw password
>kadmin.local> addprinc SOAP/myhost.example.com -randkey
>kadmin.local>listprincs # to see the added principals
```

The Web service principal name (SOAP/myhost.example.com) is shown in the example as being created with a random password. The Web service principals use keytables (a file that stores the service principal name and key) to log into Keberos System. Using a random password increases security.

10.17.3 Configuring the Web Service Client to Use the Correct KDC

The Web service client needs to be configured to authenticate against the right KDC.

The configuration for the KDC resides at */etc/krb5.conf* for UNIX hosts, and at *C:\windows\krb5.ini* for Windows hosts.

A sample *krb5.conf* is shown in Example 10–9. Note the following:

- The file tells the kerberos run time the realm of operation and the KDC endpoint to contact.
- For Kerberos token policies to work, three additional properties need to be specified in the *libdefaults* section of this file:
  - default_tkt_enctypes
  - default_tgs_enctypes
  - permitted_enctypes

  The order of cipher suites is significant and should comply with the algorithm suite used in the client-side Kerberos policy. For example, if the KDC-supported enc-types are des3-cbc-sha1, des-cbc-md5, des-cbc-crc, arcfour-hmac, then the following order of enc-types entries should be used in client's *krb5.conf* for the following policies:

  - wss11_kerberos_with_message_protection_client_policy:
    - default_tkt_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc arcfour-hmac
    - default_tgs_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc arcfour-hmac
    - permitted_enctypes =   des3-cbc-sha1 des-cbc-md5 des-cbc-crc arcfour-hmac
  - wss11_kerberos_with_message_protection_basic128_client_policy:

- * default_tkt_enctypes = arcfour-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc
- * default_tgs_enctypes = arcfour-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc
- * permitted_enctypes = arcfour-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc

Example 10–9 Sample krb5.conf File

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = example.com
dns_lookup_realm = false
dns_lookup_kdc = false
default_tkt_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc arcfour-hmac
default_tgs_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc arcfour-hmac
permitted_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc arcfour-hmac

[realms]
example.com =
{kdc = exampleadminserver.com:88  admin_server = exampleadminserver.com:749


default_domain = us.example.com  }
[domain_realm]
us.example.com = example.com

[kdc]
profile = /var/kerberos/krb5kdc/kdc.conf

[appdefaults]
pam =
{   debug = false    ticket_lifetime = 36000   renew_lifetime = 36000


forwardable = true    krb4_convert = false  }
```

10.17.4 Setting the Service Principal Name In the Web Service Client

The Web service client that is enforcing Kerberos client-side policies needs to know the service principal name of the service it is trying to access. You set the service principal name in "Creating Principals" on page 10-86.

You can specify a value for *service.principal.name* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The default (place holder) value is *HOST/localhost@example.com*.

10.17.5 Setting the Service Principal Name In the Web Service Client at Design Time

The Web service client that is enforcing Kerberos client-side policies needs to know the service principal name of the service it is trying to access. You set the service principal name in "Creating Principals" on page 10-86.

Use a configuration override to specify the service principal name at design time, as follows:

```
JAX-WS Clients:
((BindingProvider)port).getRequestContext().put(SecurityConstants.ClientConstants.
WSSEC_KERBEROS_SERVICE_PRINCIPAL,
SOAP/myhost.example.com@example.com);
```

10.17.6 Configuring the Web Service to Use the Correct KDC

Configure the Web service to authenticate against the correct KDC. The configuration for the KDC resides at */etc/krb5.conf* for UNIX hosts, and at *C:\windows\krb5.ini* for Windows hosts.

A sample KDC configuration for a Web service client is shown in Example 10–9. This example also applies to the Web service KDC configuration.

10.17.7 Using the Correct Keytab File in Enterprise Manager

To use the correct keytab file, you

- Extract and install the keytab File
- Modify the krb5 login module

These tasks are described in the sections that follow.

10.17.7.1 Extract and Export the Keytab File

Extract the key table file, which is often referred to as the keytab, for the service principal account from the KDC and install on the machine where the Web service implementation is hosted.

For example. you can use a tool such as *kadmin.local* to extract the keytab for the service principal name, as follows:

```
>kadmin.local>ktadd -k /tmp/krb5.keytab SOAP/myhost.example.com
```

Export the keytab file to the machine where the Web service is hosted. The keytab is a binary file; if you ftp it, use binary mode.

10.17.7.2 Modify the krb5 Login Module to use the Keytab File

Modify the krb5 login module as described in "Configuring the SAML and Kerberos Login Modules" on page 10-60 to identify the location of the Web service KDC file.

For example, assume that the keytab file is installed at */scratch/myhome/krb5.keytab*. Note the changes for the keytab and principal properties:

- principal value=SOAP/myhost.example.com@example.com
- useKeyTab value=true
- storeKey value=true
- keyTab value=/scratch/myhome/krb5.keytab
- doNotPrompt value=true

10.17.8 Authenticating the User Corresponding to the Service Principal

The Web services run time must be able to verify the validity of the kerberos token.

If the token is valid, Oracle Platform Security Services (OPSS) must then be able to authenticate the user corresponding to the service principal against one of the configured WebLogic Server Authentication providers. (Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.)

The user must therefore exist and be valid in the identity store used by the Authentication provider.

For example, consider a service principal such as *SOAP/myhost.example.com@example.com*. In this example, a user with the name *SOAP/myhost.example.com* must exist in the identity store. Note that *@domain* should not be part of your user entry.

10.17.9 Creating a Ticket Cache for the Web Service Client

Perform these steps to create a ticket cache for the Web service client:

1. Log in to the Kerberos system using the user principal you created for the client.

   ```
   >kinit fmwadmin password
   ```

2. This creates a ticket cache on the file system with ticket granting ticket. To see this:

   ```
   >klist -e
   ```
   Information similar to the following is displayed:

   ```
   Credentials cache: /tmp/krb5cc_36687
   Default principal: fmwadmin@example.com, 1 entry found.
   [1]  Service Principal:  krbtgt/example.com@example.com
        Valid starting:  Sep 28, 2007 17:20
        Expires:         Sep 29, 2007 17:20
            Encryption type: DES3 CBC mode with SHA1-KD
   ```

   Make sure the encryption type reflects what is shown above.

3. Run the Web service client.

Alternatively, you can run the Web service client without first logging into the Kerberos. You are prompted for the Kerberos user name and password. Note that in this case a ticket cache is not created on the file system; it is maintained in memory.

10.17.10 Configuring Kerberos With SPNEGO Negotiation

SPNEGO (Simple and Protected GSSAPI Negotiation Mechanism) is a standard that enables a client and a service to negotiate a method to use for authentication. Because SPNEGO uses HTTP headers to perform the negotiation, it is especially useful in a cross-platform context such as the web, where SOAP and REST endpoints that use HTTP are common.

When Kerberos is used in SPNEGO negotiation, the Kerberos token is wrapped in the HTTP header under the auth-scheme Negotiate. The WWW-Authenticate and Authorization headers are used to communicate the SPNEGO token between the client and the service, as follows:

1. The client requests access to a protected service on the server without any Authorization Header.

2. Since there is no Authorization Header in the request, server responds with the status code 401 (Unauthorized) and WWW-Authenticate: Negotiate.

3. The client uses the user credentials to obtain the Kerberos token and then sends it to the server in the Authorization header of the new request. For example, Authorization: Negotiate a87421000000492aa874209....
4. The server decodes this token by passing it to the acceptSecContext() GSS-API. If the context is not complete (in the case of Mutual Authentication) the server responds with a 401 status code and a WWW-Authenticate header containing the GSS-API data. For example, WWW-Authenticate: Negotiate 74900a2a....
5. The client decodes this data and sends new data back to the server. This cycle will continue until the security context is established.

Oracle WSM provides the following assertion templates to enable clients and services using SPNEGO negotiation to use Kerberos for authentication:

- oracle/http_spnego_token_client_template
- oracle/http_spnego_token_service_template

These assertion templates can be used by policies attached to SOAP or REST endpoints.

10.18 Using Active Directory with Kerberos and Message Protection

You can use Microsoft Active Directory with the Key Distribution Center (KDC) as your KDC. This section describes how to configure the KDC through Active Directory for use with Kerberos and message protection.

This section assumes that you are already familiar with Active Directory. See your Active Directory documentation for additional details.

10.18.1 Setting Up the Web Service Client

This section describes the following tasks:

- "Create a User Account" on page 10-90
- "Create a Keytab File" on page 10-90
- "Set the Service Principal Name" on page 10-91

10.18.1.1 Create a User Account

Use Active Directory to create a new user account.   Do not use DES encryption. By default, the user account is created with RC4-HMAC.

For example, you might create a user `testpol` with the user logon name `test/testpol`.

The user logon name should be of the form `container/name`. You can create the account in any container.

10.18.1.2 Create a Keytab File

Use `ktpass` to create a keytab file:

```
ktpass -princ test/testpol@{domain} -pass {...}  -mapuser testpol -out
 testpol.keytab -ptype KRB5_NT_PRINCIPAL  -target {domain}
```

where `test/testpol` is the Service Principal Name and it is mapped to the user `testpol`. Do not set `/desonly` or crypto as `des-cbc-crc`.

10.18.1.3 Set the Service Principal Name

Use `setSpn` to map the Service Principal Name to the user:

```
setSpn -A test/testpol testpol
setSpn -L testpol (this should display the availabel mapping)
```

There should be only one Service Principal Name mapped to the user. If there are multiple Service Principal Names mapped to the user, remove them using `setSpn -D <spname> <username>`.

10.18.2 Set Up the Web Service

Perform these steps to set up the Web service:

1. Attach the Kerberos policy to your Web service.
2. Configure the Web service client to authenticate against the right KDC.

   The configuration for the KDC resides at `/etc/krb5.conf` for UNIX hosts, and at `C:\windows\krb5.ini` for Windows hosts.

   Configure the default domain and realm in the `krb5.conf` or `krb5.ini` file. Enable the `RC4-HMAC` encryption type (available in JDK6).

   ```
   [libdefaults]

   default_tkt_enctypes = rc4-hmac
   default_tgs_enctypes = rc4-hmac
   permitted_enctypes = rc4-hmac
   ```

3. Export the keytab file you created in "Create a Keytab File" on page 10-90 to the system where the Web service is hosted. The keytab is a binary file; if you ftp it, use binary mode.
4. Verify the keytab file using kinit:

   ```
   kinit -k -t <absolute path the the keytab file> <Service Principal Name>
   ```

5. Modify the krb5 login module as described in "Configuring the SAML and Kerberos Login Modules" on page 10-60 to specify the keytab location and the Service Principal Name.

   Use the absolute path to the keytab file. Also, be sure to add `@realmname` to the Service Principal Name. For example:

   ```
   principal value=test/testpol@example.com
   ```

10.19 SAML Message Protection Use Case

Assume that you have a Web service client that you want to protect with the wss11_saml_token_with_message_protection_client_policy policy, and a corresponding Web service that you want to protect with the wss11_saml_token_with_message_protection_service_policy policy.

This section steps through the procedure for using these two policies.

The following topics are described:

- "What You Need to Know" on page 10-92
  - "Requirements of the wss11_saml_token_with_message_protection_service_policy" on page 10-92

10.19.1 What You Need to Know

This section describes what you need to know to configure this SAML message protection use case. The following topics are described:

10.19.1.1 Requirements of the wss11_saml_token_with_message_protection_service_policy

wss11_saml_token_with_message_protection_service_policy enforces message-level protection (that is, message integrity and message confidentiality), and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

Messages are protected using WS-Security's Basic 128 suite of symmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

Therefore, when you use the keytool (or other tool) to create the signature and encryption keys needed by this policy, you need to make sure you use the RSA key mechanism, the SHA-1 algorithm, and AES-128 bit encryption to satisfy the policy requirements for the key.

10.19.1.2 How Are Messages Protected Via Symmetric Keys?

This policy uses symmetric key technology. Symmetric key cryptography relies on a single, shared secret key, as follows:

1. The client creates the symmetric key, uses it to sign and encrypt the message, and shares it with the Web service in the request message.

To protect the symmetric key, the symmetric key sent in the request message is encrypted using the service's certificate.

2. The Web service uses the symmetric key in the request message to verify the signature of the request message and decrypt it, and to then sign and encrypt the response message.

Consider the following process flow.

To create the request, the Oracle WSM agent does the following:

1. Generates the shared symmetric key and uses it to both sign and encrypt the request message.
2. Uses its own private key to "endorse" the signature of the request message.
3. Uses the Web service's public key to encrypt the symmetric key.
4. Sends the symmetric key along with the request to the Web service. The client sends its public key in the request so that the Web service can verify the endorsement.

When the Web service gets the request, it does the following:

1. Uses its private key to decrypt the symmetric key.
2. Uses the symmetric key to decrypt the request message and to verify its signature.
3. Uses the client's public key in the request message to verify the endorsement signature.

To send the response back to the client, the Web service does the following:

1. Uses the same client-generated symmetric key sent along with the request to sign the response message.
2. Uses the same client-generated symmetric key to encrypt the response message.

When the Oracle WSM agent receives the response message, it does the following:

1. Uses the symmetric key it generated initially to decrypt the response message.
2. Uses the symmetric key it generated initially to verify signature of the response message.

10.19.1.3 What Keys Must Be in the Keystore?

If the client and Web service are in the same domain with access to the same keystore, they can share the same private/public key pair.

That is, the client can use the private key "orakey" to endorse the signature of the request message and the public key "orakey" to encrypt the symmetric key. The Web service in turn uses the public key "orakey" to verify the endorsement, and the private key "orakey" to decrypt the symmetric key.

For demonstration purposes, this use case creates one key pair.

10.19.1.4 Multi-Domain Use Case (Keystore Hardening)

If the client and Web service are not in the same domain and do not have access to the same keystore, the client and Web service must each have a private/public key pair.

Consider the following requirements in a multiple-domain use case, as shown in Table 10–3.

Table 10–3 Multiple-Domain Use Case Requirements

Web Service Client	Web Service
Needs its own private/public key pair in the client keystore.	Needs its own private/public key pair in the service keystore.
Needs the Web service public key.	Needs the intermediary and root certificate corresponding to the client's public key in the keystore. These certificates will be used to verify the signature by generating a trusted certificate chain.
Generates symmetric key at run time	Needs the symmetric key, but this is sent in the request message.

For the public key the client uses to encrypt the symmetric key -- that is, the public key of the Web service -- you have two approaches:

- The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57. Therefore, in this use the Web service's public key does not have to be in the client's keystore.
- If the certificates is not published in the WSDL, you can specify a value for `keystore.recipient.alias` on the Configurations page, or override it on a per-client basis using the Security Configuration Details control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages. In this approach, the Web service's public key must be in the client's keystore.

10.19.1.5 When to Override the SAML Issuer

The saml.issuer.name property of the client policy identifies the issuer of the SAML token, and defaults to a value of `www.oracle.com`. This use case uses the `www.oracle.com` default.

You can optionally specify a value for saml.issuer.name on the Configurations page, or override it on a per-client basis using the Security Configuration Details control when you attach the policy.

If you do use a different SAML authority (issuer) in the policy, that issuer name must be configured in the client and included in the list of possible issuers in the SAML login module. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for information on how to do this.

10.19.2 Main Steps

This section describes the steps you follow to configure the SAML message protection use case. The following topics are described:

- "Create a WebLogic Server User" on page 10-95
- "Create a Java Keystore" on page 10-96
- "Configure the Web Services Manager Keystore" on page 10-96
- "Store the Password for the Decryption Key in the Credential Store" on page 10-97
- "Attach the Policy to Your Web Service" on page 10-97

- "Attach the Policy to Your Web Service Client" on page 10-97

10.19.2.1 Create a WebLogic Server User

The user in the SAML token must already exist in the WebLogic Server identity store.

The Web service run time extracts the SAML token from the WS-Security header and uses the name in the SAML token to validate the user against the WebLogic Server identity store.

Specifically, the SAML login module (see "Configuring the SAML and Kerberos Login Modules" on page 10-60 verifies the SAML tokens on behalf of the Web service. The SAML login module then extracts the username from the verified token and (indirectly) passes it to Oracle Platform Security Services (OPSS) to complete the authentication.

Any configured WebLogic Server authentication provider can then be invoked, including the default Authentication provider.

Create the User

You use the WebLogic Server Administration Console to add the user to the identity store, as described in the *Oracle WebLogic Server Administration Console Help*.

The steps are repeated here for ease of use.

To create a user in the WebLogic Server Administration Console:

1. In the left pane select **Security Realms**.
2. On the Summary of Security Realms page select the name of the realm (for example, myrealm).
3. On the Settings for Realm Name page select **Users and Groups** and then **Users**.

   The User table displays the names of all users defined in the Authentication provider.
4. Click **New**.
5. In the Name field of the Create New User page enter the name of the user.

   User names are case sensitive and must be unique. Do not use commas, tabs or any other characters in the following comma-separated list: <>, #, |, &, ?, ( ), { }
6. (Optional) In the Description field, enter a description. The description might be the user's full name.
7. In the Provider drop-down list, select the Authentication provider for the user.

   If multiple WebLogic Authentication providers are configured in the security realm, they will appear in the list. Select which WebLogic Authentication provider's database should store information for the new user.
8. In the Password field, enter a password for the user.

   The minimum password length for a user defined in the WebLogic Authentication provider is 8 characters.
9. Re-enter the password for the user in the Confirm Password field.
10. Click **OK** to save your changes.

    The user name appears in the User table.

10.19.2.2 Create a Java Keystore

This section provides an outline of how to create and manage the Java keystore with the keytool utility. It describes how to create a keystore and load the private key and trusted CA certificates.

You can find more detailed information on the commands and arguments for the keytool utility at the following Web address: http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html.

> **Note:** You specify an alias when you add an entity to the keystore using the -genkey command to generate a key pair (public and private key), or when you use the -import command to add a certificate or certificate chain to the list of trusted certificates.
>
> Subsequent keytool commands must use this same alias to refer to the entity.

1. Create a new key pair and self-signed certificate.

   Use the genKey command to create the key pair (public and private key). genKey creates a new private key if one does not exist.

   The following command generates an RSA key, with RSA-SHA1 as the signature algorithm, with the alias "orakey" in the default-keystore.jks keystore. You can choose any alias name; you do not need to name your alias "orakey".

   ```
   keytool -genkey -alias orakey -keyalg "RSA" -sigalg "SHA1withRSA" -dname
   "CN=test, C=US" -keystore default-keystore.jks
   ```

   The keytool utility prompts for the needed key and keystore passwords. You need these passwords later.

2. Generate a certificate request to the certificate authority.

   Use the -certreq command to generate the request. The following commands generates a certificate request for the orakey alias.

   The CA will return a certificate or a certificate chain.

   ```
   keytool -certreq -alias orakey -sigalg "SHA1withRSA" -file certreq_file
   -storetype jks -keystore default-keystore.jks
   ```

3. Replace (import) the self-signed certificate with the trusted CA certificate.

   You must replace the existing self-signed certificate with the certificate returned from the CA. To do this, use the -import command. The following command replaces the trusted CA certificate in the default-keystore.jks keystore. The keytool utility prompts for the needed password.

   ```
   keytool -import -alias orakey -file certreq_file -keystore default-keystore.jks
   ```

10.19.2.3 Configure the Web Services Manager Keystore

Perform these steps to configure the Oracle Web Services Manager keystore:

1. In the navigator pane, expand WebLogic Domain to show the domain for which you need to configure the keystore. Select the domain.

2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Security**, and then **Security Provider Configuration**.

   Click the plus sign (+) to expand the Keystore control near the bottom of the page, then click Configure.

   The Web Services Manager Keystore Configuration page is displayed, as shown in Figure 10–3.

3. If it is not already enabled, click the **Configure Keystore Management** check box.

4. Enter the path and name for the keystore that you created. By default, the keystore name is default-keystore.jks, as used in this use case. The keystore type must be JKS.

5. Enter the password for the keystore and confirm it.

6. Enter the alias and password for the signature and encryption keys.

   In this use case, orakey is the alias for both the signature and encryption keys.

   Confirm the passwords.

7. Click **OK** to submit the changes.

   Note that all fields on this page require a restart of Fusion Middleware Control to take effect.

10.19.2.4 Store the Password for the Decryption Key in the Credential Store

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use *keystore.enc.csf.key* as the key name.

10.19.2.5 Attach the Policy to Your Web Service

Attach wss11_saml_token_with_message_protection_service_policy to your Web service as described in "Attaching a Policy to a Single Subject" on page 8-3.

Configure the policy assertion for message signing and message encryption.

The default is to sign and encrypt the entire body for the request the response. You have the option to not do this and to instead specify the specific body elements that you want to sign and encrypt. You can also additionally specify header elements that you want to sign and encrypt. Whatever you set here mush match the client policy settings.

Note: You can override keystore.sig.csf.key and keystore.enc.csf.key, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

If you do override these values, the keys for the new values must be in the keystore. That is, overriding the values does not free you from the requirement of configuring these keys in the keystores.

10.19.2.6 Attach the Policy to Your Web Service Client

Attach wss11_saml_token_with_message_protection_client_policy to your Web service client, as described in "Attaching Policies to Web Service Clients" on page 8-11.

Configure the policy assertion for message signing, message encryption, or both.

The default is to sign and encrypt the entire body. You have the option to not do this and to instead specify the specific body elements that you want to sign and encrypt. You can also additionally specify header elements that you want to sign and encrypt. Whatever you set here must match the Web service policy settings.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57. The certificate in the WSDL is the service's public key by default, as determined by the encryption key you specified ("orakey") when you configured the Web Services Manager keystore.

Therefore, you do not need to set or change `keystore.recipient.alias`.

You can optionally specify a value for `saml.issuer.name` on the Configurations page, or override it on a per-client basis using the Security Configuration Details control when you attach the policy. The `saml.issuer.name` property defaults to a value of `www.oracle.com`. See "When to Override the SAML Issuer" on page 10-94.

You can specify a value for `user.roles.include` on the Configurations page, or override it on a per-client basis using the Security Configuration Details control when you attach the policy.

10.20 WS-Trust Policies and Configuration Steps

This section describes the predefined WS-Trust policies and how to configure and use them. The following topics are described:

- "Overview of Web Services WS-Trust" on page 10-98
- "Setting Up Automatic Policy Configuration for STS" on page 10-104
- "Programmatic Configuration Overrides for WS-Trust Client Policies" on page 10-110
- "Supported STS Servers" on page 10-111
- "Available WS-Trust Policies" on page 10-109

10.20.1 Overview of Web Services WS-Trust

The WS-Trust 1.3 specification defines extensions to WS-Security that provide a framework for requesting and issuing security tokens, and to broker trust relationships. WS-Trust extensions provide methods for issuing, renewing, and validating security tokens.

To secure communication between a Web service client and a Web service, the two parties must exchange security credentials. As defined in the WS-Trust specification, these credentials can be obtained from a trusted `SecurityTokenService` (STS), which acts as trust broker. That is, the STS must be trusted by both the Web service client and the Web service to provide interoperable security tokens.

This section describes the following topics:

- "How the STS Configuration is Obtained" on page 10-99
- "Typical Token Request and Response" on page 10-99
- "Example WS-Trust Use Case" on page 10-100
- "Token Lifetime" on page 10-101
- "What Token Types Are Exchanged?" on page 10-101

- "Overview of Sender Vouches in WS-Trust" on page 10-104

10.20.1.1 How the STS Configuration is Obtained

Typically, your environment will have only one STS. If you have a hundred different Web services, all of which have attached this STS config policy, you can easily change all of your Web services to point to a different STS by changing the policy.

The STS is also a Web service. To communicate with the STS, the client application needs to know the STS details, such as the port-uri, port-endpoint, wsdl-uri, and the security tokens it can accept from clients trying to authenticate to it.

There are two mechanisms by which STS information becomes available to the client.

- Automatic (Client STS) Policy Configuration (see "Setting Up Automatic Policy Configuration for STS" on page 10-104) is involved. Automatic Policy Configuration dynamically generates the information about the STS by parsing the STS WSDL document.

  Automatic Policy Configuration is triggered when the STS config policy is attached to the Web service and not the client. Additionally, the only information provided in the STS config policy is the port-uri of the target STS.

  When this policy is attached to the Web service along with the issued token service policy, the port-uri of the STS appears as the Issuer-Address in the IssuedToken assertion of the Web service WSDL. As a result, all the other STS information (target namespace, service name, endpoint, and so forth) is obtained by accessing the STS WSDL and is saved in memory as the STS config. This information is stored only in memory and is not persisted in the Oracle WSM Repository. (For details about the repository, see Chapter 17, "Maintaining the Oracle WSM Repository."

  If you specify the STS URI in the Web service STS config policy and attach it to the Web service, the client is forced to use that STS; it cannot override it.

- You do not use Automatic Policy Configuration and instead attach the STS config policy to the client and specify all the STS-related information (port-endpoint, port-uri, public key alias, a reference to an Oracle WSM client policy to be used for authenticating to the STS) before invoking the Web service. In this case, all the information is already available to the run time from the STS config policy.

10.20.1.2 Typical Token Request and Response

The general token request/response process works as follows. These steps are explained further in the use case described in "Example WS-Trust Use Case" on page 10-100.

1. The Web service client wants to invoke a Web service. The Oracle WSM agent attempts to fetch the WSDL of the Web service and extract the issued token service policy. The Oracle WSM agent uses the local client policy (as optionally overridden) to talk to the STS identified in the WSDL.

   The Web service policy can require the issued token to be from a specific STS.

2. The Web service client requests that the STS issue a token. The Web service client can request the token from a specific STS.

   The Request Security Token (RST) is a request for a security token. The RequestSecurityTokenResponse (RSTR) is a response generated by the STS in response to the RST with claims for the requested user.

3. The Web service client processes the RSTR sent by the STS and propagates the issued token to the Web service.
4. The Web service processes and verifies the issued token and generates a response back.

10.20.1.3 Example WS-Trust Use Case

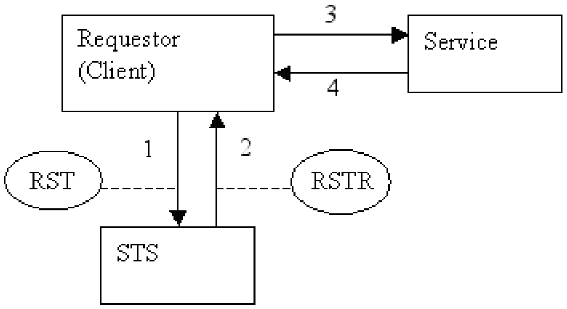
This section describes a sample use case for WS-Trust.

1. The Web service client invokes a Web service. The WSDL for the Web service indicates that the Web service requires a security token from a specific STS.
2. The Web service client (the requestor) sends an authentication request, with accompanying credentials, to the STS.
3. The STS verifies the credentials presented by the client, and then in response issues a security token that provides proof that the client has authenticated with the STS. The response message RSTR has the token and (optionally) claims for the authenticated user.
4. The requestor verifies the RSTR, extracts the token, and passes it to the Web service.
5. The Web service receives the issued token and verifies that the token was issued by a trusted STS. This proves that the client has successfully authenticated with the STS.

   Once the token is validated, the Web service processes the request and responds back.

Figure 10–30 illustrates the message flows between the requestor, STS, and the Web service.

Figure 10–30 STS Use Case Message Flow



10.20.1.4 On Behalf Of Use Cases

"On Behalf Of" is an identity propagation use case, in which the Web service client requests the STS token on behalf of another entity.

Consider the following scenario:

1. The Web service client invokes the STS to get a token for another entity. This entity can be the end user or any other external entity. The entity's credentials are included in the RST in the `onBehalfOf` element.
2. The STS verifies the credentials presented by the Web service client and issues a security token for the entity identified in the `onBehalfOf` element.

3. The Web service client verifies the RSTR, extracts the token, and passes it to the Web service.
4. The Web service receives the SAML assertion for the end user and verifies that the token was issued by a trusted STS.

The "On Behalf Of" use case relies on the `sts.auth.on.behalf.of.csf.key` and `on.behalf.of` properties described in Table 8–4. If the "On Behalf Of" username is obtained from the Subject, it is a username without a password.

If `sts.auth.on.behalf.of.csf.key` identifies a CSF key for the "On Behalf Of" user entity, the identity established using that CSF key is sent on behalf of the other entity. It can be a username with or without a password.

10.20.1.5 Token Lifetime

The RSTR response message from an STS may contain a lifetime element (`<trust:Lifetime>`) indicating the validity of the returned token. If the lifetime element is present, Oracle WSM validates the timestamp and rejects the message if the response has expired.

10.20.1.6 What Token Types Are Exchanged?

Although an STS can theoretically receive any token from the client and exchange it for any other token, in practice the STS generally accepts one of the following tokens and returns a SAML assertion:

- Username token. For this token type:

  1. The Web service client sends a user name and password to the STS.
  2. The STS verifies the password and returns a SAML assertion.
  3. The client sends the SAML assertion to the Web service.

  This scenario is useful when the Web service does not have the ability to verify passwords, so it relies on the STS to verify them.

- Kerberos token. For this token type:

  1. The client sends a user name and password to a KDC and gets a Kerberos token.
  2. The client sends the Kerberos token to the STS and gets a SAML assertion.
  3. The client sends the SAML assertion to the Web service.

  This scenario is useful in Windows environments. Clients running on the Windows machine have the logged-on user context, and they can use this context to get a SAML assertion from the STS for that user.

  In this scenario, the clients do not have the password so they cannot use a username token, they can use only Kerberos.

- X509 token -- For this token type the client uses a private key to authenticate itself to the STS.

In response, the STS generally returns one of the following tokens:

- SAML Holder of Key Symmetric. The SAML assertion that is returned by the STS is meant only for the particular client that sent its client token (username token, Kerberos, X509, etc) to the STS.

  A rogue client should not be allowed to steal this SAML assertion and use it. This is accomplished by a "proof key," which can be either symmetric or asymmetric.

A symmetric proof key is generated on the STS side, or on the client side, or by taking inputs from both sides, as described in "How the Proof Key is Determined (SAML HOK Only)" on page 10-103.

The STS puts this symmetric proof key in the SAML HOK assertion in an encrypted form that only the Web service can decrypt. Then, it signs the entire SAML assertion (including the encrypted proof key) and sends it to the client.

When the client sends this SAML assertion to the server, it also needs to sign something with this proof key. The Web service will at first verify the STS signature of the SAML assertion, extract the proof key from the SAML assertion, and then decrypt it and verify the client's signature. This client's signature "proves" to the server that the client has the proof key.

Because this proof key is never sent in clear text, a rogue client cannot get it by network sniffing. Even if a rogue client gets the SAML assertion by network sniffing, it cannot make use of it, because it does not have the proof key and cannot sign with it. Therefore, the rogue client cannot prove to the server that it is allowed to use the SAML assertion.

- SAML Holder of Key Asymmetric. The asymmetric proof key works as follows.

  1. The client generates a public/private key pair.
  2. It keeps the private key and securely sends the public key to the STS along with its token (username token, Kerberos, X509, and so forth.)
  3. The STS verifies the client's token, and returns a SAML assertion containing the public key. The entire SAML assertion (including the public key) is signed by the STS and returned to the client.
  4. The client then sends a SAML HOK asymmetric assertion to a Web service, and it signs something with the private key of that public-private key pair.
  5. The Web service verifies the STS's signature of the SAML assertion, then extracts the public key from the SAML assertion and uses it to verify the client's signature.

     This client's signature proves to the Web service that the SAML assertion is being used correctly, and was not stolen and replayed.

Note: Unlike in the case of SAML HOK symmetric key, this public key in SAML HOK is not encrypted. This reduces the amount of configuration required on the STS side.

For SAML HOK symmetric, the STS must be configured with each Web service's certificate so that it can encrypt the symmetric key for that Web service. This is not required for SAML HOK asymmetric.

Also, the same SAML HOK asymmetric token can be sent to any Web service because it is not encrypted with a particular Web service's key.

> **Note:** Even though there is a public/private key pair, there is no certificate involved. That is, the public key is not sent to a Certificate Authority to request a certificate.
>
> Instead, the STS acts similar to a CA. A CA takes in a public key and returns a certificate. In this case, the STS takes in a public key and returns a SAML assertion.
>
> However, unlike a certificate whose lifetime is usually in many years, the SAML assertion issued by the STS usually has a lifetime of a few hours, after which the client would have to generate a new key pair and request a new SAML assertion.
>
> Because of this short life, there is no need for the revocation checking that is required for certificates. This makes it attractive on the client side, because there are no client keys to manage.

- SAML Bearer -- The SAML bearer key has no proof key associated with it. Therefore, it must be used over SSL to prevent any rogue client from stealing and replaying it.

10.20.1.6.1 How the Proof Key is Determined (SAML HOK Only) For SAML Holder of Key (HOK), a proof key is required to protect communications between the client and the Web service. The proof key indicates proof of possession of the token associated with the requested security token.

You specify the requirements for the proof key type in the `oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy` in the `<key-type>` entry in the `<sp:IssuedToken>` policy assertion. For example,

```
<orasp:request-security-token-template
orasp:key-type = "Symmetric"
```

or

```
orasp:key-type = "Public"
```

Symmetric, asymmetric, and no proof key (not defined) are supported.

These possible values of `<key-type>` are contained in the WS-Trust 1.3 specifications:

- `http://docs.oasis-open.org/ws-sx/ws-trust/200512/PublicKey`
- `http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey`

10.20.1.6.2 Calculating a Symmetric Proof Key If a symmetric proof key is required by the Web service's security policy, the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.

However, the STS determines what to use for the proof key. When processing the token request, the STS can:

- Accept the client entropy as the sole key material for the proof key. In this case, there is no `<wst:RequestedProofToken>` element present in RSTR; the proof key is implied.

  The Oracle WSM agent uses the client entropy as the key material for signing and encryption.

- Accept the client entropy as partial key material and contribute additional STS server-side entropy as partial key material to compute the proof key as a function of both partial key materials.

  There is a `<wst:Entropy>` element containing the STS-supplied entropy in the RSTR. The `<wst:RequestedProofToken>` element is also present in RSTR and it contains the computed key mechanism. The default value for the algorithm is `http://docs.oasis-open.org/ws-sx/ws-trust/200512/CK/PSHA1`.

  The Oracle WSM agent and the STS compute the proof key by combining both entropies using the specified computed key mechanism.

- Reject the client-side entropy and use the STS server-side entropy as the sole key material for the proof key.

  There is a `<wst:RequestedProofToken>` element present in RSTR that contains the proof key. The Oracle WSM agent uses the STS entropy as the key material for signing and encryption.

10.20.1.6.3 Requesting an Asymmetric Proof Key An asymmetric proof key uses private/public key pairs, and is termed "asymmetric" because the public and private keys are different.

When requesting an asymmetric key token, the RST includes the `wst:KeyType` element with the following URI:
`http://docs.oasis-open.org/ws-sx/ws-trust/200512/PublicKey`.

10.20.1.7 Overview of Sender Vouches in WS-Trust

An STS typically returns a SAML HOK or SAML Bearer token. However, an STS can also return SAML sender vouches tokens.

SAML sender vouches has a completely different trust model. In HOK and Bearer the the SAML assertion is issued by an STS and is signed by the STS. In this case, the Web service does not trust the client directly, but it trusts the STS. When the Web service receives an HOK or Bearer token, it verifies the signature against the trusted STS.

This indirect trust model greatly simplifies the trust store management. That is, if there are five clients talking to five Web services using message protection, then each of the Web services must know the five client public keys. Therefore, if there an STS in between, the Web services need to know only the public key of the STS.

For SAML sender vouches, the Web service trusts the client directly. A SAML sender vouches token is typically directly generated by a client and signed by the client private key. However a client may choose to ask the STS to generate the token. The STS does not sign the SAML assertion in this case, and simply returns it to the client. The client signs the SAML sender vouches token as before and sends it to the Web service. The Web service is not aware that the client obtained the SAML sender vouches token from an STS and it checks the client signature.

10.20.2 Setting Up Automatic Policy Configuration for STS

Automatic Policy Configuration dynamically generates the information about the STS by parsing the STS WSDL document.

When the STS config policy is attached to the Web service (and not to the client) Automatic Policy Configuration happens at run time on the first connect from client to server.

The only information you provide in the STS config policy (`oracle/sts_trust_config_service_policy`) is the port-uri of the target STS. When this policy is attached to the Web service (along with the issued token service policy) the port-uri of the STS appears as the Issuer-Address in the IssuedToken assertion of the Web service WSDL.

As a result, Oracle WSM obtains the other STS information (target namespace, service name, endpoint, and so forth) by accessing the STS WSDL and is saved in memory as the STS config. This information is saved in memory but is not persisted in MDS.

This section describes the following topics:

- "Requirements for Automatic Policy Configuration" on page 10-105
- "Setting Up Automatic Policy Configuration: Main Steps" on page 10-105
- "Manually Configuring the STS Config Policy From the Web Service Client: Main Steps" on page 10-107

10.20.2.1 Requirements for Automatic Policy Configuration

There are several requirements for successfully communicating with the STS using Automatic Policy Configuration:

- Automatic Policy Configuration does not work with JSE clients. If you are using JSE clients in a WS-TRUST scenario, you need to provide all the STS configuration information to the client by attaching both the sts_trust_config_client_policy and the issued token client policy.
- The `oracle/sts_trust_config_service_policy` policy must be attached to the Web service. If it is not, you cannot use Automatic Policy Configuration and must instead manually configure the `oracle/sts_trust_config_client_policy` policy for the client, as described in "Manually Configuring the STS Config Policy From the Web Service Client: Main Steps" on page 10-107.
- Automatic Policy Configuration cannot be used for SAML sender vouches confirmation because the trust is between the Web service and the client. The Web service WSDL will not have any information about the STS.
- The certificate and public key alias of the STS must be in the keystore. The default alias name is `sts-csf-key`. See "Configuring Keystores for Message Protection" on page 10-9 for information on how to do this.
- The client's public key must be available in the STS keystore.

10.20.2.2 Setting Up Automatic Policy Configuration: Main Steps

Perform these steps to use Automatic Policy Configuration.

1. "Configure a Policy for Automatic Policy Configuration" on page 10-105
2. "Configure a Web Service Client for Automatic Policy Configuration" on page 10-106
3. "Configure a Web Service for Automatic Policy Configuration" on page 10-107

Configure a Policy for Automatic Policy Configuration

Perform these steps to configure a policy for automatic policy configuration:

1. Decide which STS your Web service trusts and import that STS's public certificate into the Oracle WSM keystore.
2. Optionally, add the DN of the STS to the Trusted STS list, as described in "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23.

3. If you want to use SAML HOK symmetric, you need to add an entry in the Oracle OpenSSO STS configuration for your Web service and the certificate of your Web service. The STS encrypts symmetric keys using this certificate.
4. Make a copy of the `sts_trust_config_service_policy` policy.
5. Edit the `orasp:port-uri` field to add the port-uri of the STS.

   An STS usually exposes multiple URI points for different input and output token types; use the URI corresponding to the token that you want. For Oracle OpenSSO STS, the possible values for `orasp:port-uri` are as follows:

   - http://<host:port>/openssosts/sts/wss10x509
   - http://<host:port>/openssosts/sts/wss10un
   - http://<host:port>/openssosts/sts/wss11kerberos
   - https://<host:ssl_port>/openssosts/sts/tlswss10un

Configure a Web Service Client for Automatic Policy Configuration

Perform these steps to configure a Web service client for automatic policy configuration:

1. Attach the issued token policy to your Web service client, depending on what type of token the Web service requires.

   The following predefined issued token policies are provided:

   - `oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy` for SAML HOK.
   - `oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy` for SAML Bearer.
2. Set or override the following properties of the issued token policy depending on the use case. See Table 8–4 for the property descriptions.

   - `sts.auth.user.csf.key`
   - `sts.auth.x509.csf.key`
   - `sts.keystore.recipient.alias`
   - `sts.auth.keytab.location`
   - `sts.auth.caller.principal.name`
   - `sts.auth.service.principal.name`
   - `sts.auth.on.behalf.of.csf.key`
   - `on.behalf.of`

   `sts.keystore.recipient.alias` is used for the client to STS communication for message protection and is sufficient if the client to STS communication is using wss11 message protection.

   However, if it is using wss10 message protection, you need to additionally set up the signing key and encryption key for the client, and then import the trust for these keys into the STS configuration.
3. Make sure the STS public certificate and credentials are present in the keystore and the client's public key is available in the STS keystore. See "Configuring Keystores for Message Protection" on page 10-9 for information on how to do this.

Configure a Web Service for Automatic Policy Configuration

1. Attach the edited `sts_trust_config_service_policy` to the Web service.

   > **Note:** You must attach both the `sts_trust_config_service_policy` policy and an STS issued-token service policy. The policies work as a pair.

2. Attach the issued-token service policy (corresponding to the one attached to the client). There are two predefined issued token policies:

   - `oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy` -- Use this when you want your service to accept SAML HOK asymmetric or symmetric. Do not use SSL for this policy.

     As with all other wss11 message protection policies, you must set up an encryption key.

     You can modify some options in the policy. For example, whether you want SAML 1.1 or 2.0, and whether you want asymmetric or symmetric keys.

   - `oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy` -- Use this when you want SAML Bearer. However, you must set up your Web service for SSL to use this policy.

     You can specify whether you want SAML 1.1 or 2.0.

3. Override `keystore.enc.csf.key` in the issued-token service policy, if required.

4. Make sure the client's public key is available in the Oracle WSM keystore.

10.20.2.3 Manually Configuring the STS Config Policy From the Web Service Client: Main Steps

You are encouraged to configure the STS config policy from the Web service, as described in "Setting Up Automatic Policy Configuration for STS" on page 10-104. However, in the following situations you must configure it from the Web service client:

- If you did not configure the STS config policy from the Web service, or
- If you are using the SAML sender vouches confirmation method, or
- If you are using a JSE client. Automatic Policy Configuration does not work with JSE clients.

Perform these steps to configure the STS config policy from the Web service client.

1. Optionally, use Fusion Middleware Control to create a new policy from the `oracle/sts_trust_config_template` (see "Creating a New Web Service Policy" on page 7-4) or from an existing `oracle/sts_trust_config_client_policy` policy (see "Creating a Web Service Policy from an Existing Policy" on page 7-6).

   You might find that having a unique policy makes configuration more obvious.

2. Use Fusion Middleware Control to edit your chosen `oracle/sts_trust_config_client_policy` policy.

3. The predefined `oracle/sts_trust_config_client_policy` policy is shown in Example 10–10. At a minimum, you need to provide the following information:

   - Issuer address -- `Port-uri` is the actual endpoint URI of the STS.
   - Oracle WSM security policy reference -- `policy-reference-uri` is the client policy URI that will be used by the client to communicate with the STS. The

policy you choose depends on the authentication requirements of the STS, as identified in its WSDL.

How you set this parameter determines what you must later set or override in the issued token client policy:

- If `policy-reference-uri` points to a username-based policy, then you later configure the `sts.auth.user.csf.key` parameter to authenticate to STS and create a username token. You also configure `sts.auth.x509.csf.key` to specify the signature and encryption key alias.
- If the `policy-reference-uri` points to an x509-based policy, then you later configure the `sts.auth.x509.csf.key` parameter to specify the X509 certificate for authenticating to the STS.

- port-endpoint -- This is the endpoint of the Web service, specified as `target-namespace#wsdl.endpoint(service-name/port-name).`
- Alias of STS Certificate -- `sts-keystore-recipient-alias` is the alias of the STS certificate you added to the keystore. The default alias name is `sts-csf-key`.

Example 10–10 oracle/sts_trust_config_client_policy

```
<orasp:sts-trust-config
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orasp:policy-reference-uri="oracle/wss10_username_token_with_message_protection_
client_policy"
 orasp:port-endpoint="target-namespace#wsdl.endpoint(service-name/port-name)"
 orasp:port-uri="http://host:port/sts-service" orasp:soap-version="12"
 orasp:sts-keystore-recipient-alias="sts-csf-key"
 orasp:wsdl-uri="http://host:port/sts?wsdl" orawsp:Enforced="true"
 orawsp:Silent="true" orawsp:category="security/sts-config" orawsp:name="STS Trust
Configuration">
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative" orawsp:name="StsTrustConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
<orawsp:Property orawsp:contentType="constant" orawsp:name="role"
orawsp:type="string">
<orawsp:Value>ultimateReceiver</orawsp:Value>
</orawsp:Property>
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:sts-trust-config>
```

4. Save your changes.
5. If you have not already done so, select an issued token client policy from the client policies listed in Table 10–4. Your choices are:
   - `oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy`
   - `oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy`
   - `oracle/wss11_sts_issued_saml_with_message_protection_client_policy`
6. Attach both Web service client policies to your Web service client, as described in "Attaching Client Policies Permitting Overrides" on page 8-31.   You must attach the policies in this order:

1. `sts_trust_config_client_policy`
2. Issued token client policy

   If you attach multiple instances of `oracle/sts_trust_config_client_policy`, no error is generated. However, only one instance is enforced, and you cannot control which instance that is.

7. Use Fusion Middleware Control to edit your chosen issued token client policy.
8. Save your changes.

10.20.3 Using SAML Sender Vouches with WS Trust

To set up SAML sender vouches with WS-Trust, configure the Web service without an issued token policy; that is, use the `oracle/wss11_saml_token_with_message_protection_service_policy` policy.

Configure the client with an issued token policy. Use the `oracle/wss11_sts_issued_saml_with_message_protection_client_policy`, which is meant for SAML sender vouches, and also an STS config policy.

The Automatic Policy Configuration feature (see "Setting Up Automatic Policy Configuration for STS" on page 10-104) cannot be used for SAML sender vouches because the Web service WSDL will not have information about the STS.

10.20.4 Available WS-Trust Policies

The available WS-Trust policies are listed in Table 10–4.

Table 10–4 Available WS-Trust Policies

Name	Description
oracle/sts_trust_config_service_policy	Use this policy to specify the STS configuration information that is used to invoke the STS for token exchange. You use this policy with the Web service.
oracle/sts_trust_config_client_policy	Use this policy to specify the STS configuration information that is used to invoke the STS for token exchange. You use this policy with the Web service client only when not using Automatic Policy Configuration.
oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy	This policy inserts SAML bearer assertion issued by a trusted STS. Messages are protected using SSL.
oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy	This policy authenticates users using credentials provided in SAML tokens with confirmation method bearer in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. The policy verifies that the transport protocol provides SSL message protection. This policy can be applied to any SOAP-based endpoint.
oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy	This policy inserts a SAML HOK assertion issued by a trusted STS. Messages are protected using proof key material provided by STS.
oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy	This policy authenticates a SAML HOK assertion issued by a trusted STS. Messages are protected using WS-Security's Basic 128 suite of symmetric key technologies.
oracle/wss11_sts_issued_saml_with_message_protection_client_policy	This policy inserts a SAML sender vouches assertion issued by a trusted STS. Messages are protected using the client's private key.

10.20.5 Programmatic Configuration Overrides for WS-Trust Client Policies

Table 11–2 shows the properties you can set via programmatic configuration overrides for a given policy.

Table 10–5 describes a series of sample use cases that show how to override STS properties programmatically.

Table 10–5 STS Programmatic Configuration Use Cases

Use Case	Sample Code

Token exchange username token – SAML with symmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_USER_CSF_KEY, "my-user-csf-key");`
	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY, "my-x509-csf-key");`

Token exchange x509 token – SAML with symmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY, "my-x509-csf-key");`

Token exchange username token – SAML with asymmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_USER_CSF_KEY, "my-user-csf-key");`
	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY, "my-x509-csf-key");`

Token exchange x509 token – SAML with asymmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY, "my-x509-csf-key");`
On Behalf Of token exchange with On Behalf Of username from Subject, requestor token username – SAML with symmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_USER_CSF_KEY, "my-user-csf-key");` `on.behalf.of` must be set to true.

On Behalf Of token exchange with On Behalf Of username, requestor token username – SAML with symmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_ON_BEHALF_OF_CSF_KEY, "my-on-behalf-of-csf-key");`
	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY, "my-x509-csf-key");`
	`on.behalf.of` must be set to true.

On Behalf Of token exchange with On Behalf Of username, with requestor token x509 – SAML with symmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_ON_BEHALF_OF_CSF_KEY, "my-on-behalf-of-csf-key");`
	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY, "my-x509-csf-key");`
	`on.behalf.of` must be set to true.

On Behalf Of token exchange with On Behalf Of username from Subject, with requestor token username – SAML with asymmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_USER_CSF_KEY, "my-user-csf-key");` `on.behalf.of` must be set to true.

Table 10–5 (Cont.) STS Programmatic Configuration Use Cases

Use Case	Sample Code

On Behalf Of token exchange with On Behalf Of username, with requestor token username – SAML with asymmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_ON_BEHALF_OF_CSF_KEY, "my-on-behalf-of-csf-key");`
	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_USER_CSF_KEY, "my-user-csf-key");`
	`on.behalf.of` must be set to true.

On Behalf Of token exchange with On Behalf Of username, with requestor token x509 - SAML with asymmetric proof key	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_ON_BEHALF_OF_CSF_KEY, "my-on-behalf-of-csf-key");`
	`(BindingProvider) port).getRequestContext().put(SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY, "my-x509-csf-key");`
	`on.behalf.of` must be set to true.

10.20.6 Supported STS Servers

Oracle WSM provides a standard WS-Trust client. This client has been certified to interoperate with OpenSSO STS server. To step through example scenarios using OpenSSO STS server, see "Examples Using WS-Trust with OpenSSO STS" on page 10-111.

10.21 Examples Using WS-Trust with OpenSSO STS

The following sections provide end-to-end examples using WS-Trust with Open SSO Security Token Service (STS) server to configure the following security scenarios:

- "Configuring OpenSSO STS" on page 10-111
- "SAML Holder-of-Key With Message Protection Scenario" on page 10-113
- "SAML Sender Vouches with Message Protection Scenario" on page 10-115
- "SAML Bearer with Message Protection Scenario" on page 10-117

10.21.1 Configuring OpenSSO STS

The following procedure describes the steps required to configure OpenSSO STS for use with each of the example scenarios described in this section.

1. Log in to the OpenSSO STS instance.
2. Navigate to **Configuration > Global > Security Token Service**.
3. Under Security: Security Mechanism: Security Token Accepted by STS Services enable all options.
4. Under the Credential for User Token section, add a new credential for the token with the name and password set as required.
5. Under the On Behalf of Token section, select **ldapService** from the **Authentication Chain for On Behalf of Token** drop-down list.
6. Under the Signing section, enable the following options:

   - **Is Request Signature Verified**

- **Is Response Signed Enabled** (select **Body** and **Timestamp**)

7. Under the Encryption section, enable the following options:

   - **Is Request Decrypted** (select **Body** and **Header**)

   - **Is Response Encrypted**

8. Select **AES** from the **Encryption Algorithm** drop-down list, and select **128** from the **Encryption Strength** drop-down list.

9. To support the WS-Security 1.1 Kerberos token with message protection requestor token, under the Kerberos Configuration section and configure the following values:

Table 10–6 OpenSSO STS Kerberos Token With Message Protection Configuration

Configure this property . . .	To specify . . .
Kerberos Domain Server	Fully qualified hostname of the domain server.
Kerberos Domain	Domain name.
Kerberos Service Principal	Service principal name in the following format: `<host>/<machine name>@<REALM NAME>`
Kerberos Key Tab File	Location of the key tab file created for the STS.
Is Verify Kerberos Signature	Enable only when JDK6 is used.

10. To support SSL, perform these steps:

    a. In the Token Issuance Attributes section, edit the SSL Endpoint based on your OpenSSO instance.

    b. Under Signing, enable the Disable signature validation when transport is secured with SSL option.

    c. Under Encryption, enable the Disable decryption when transport is secured with SSL option.

11. To support SSL on the server hosting the OpenSSO STS:

    - On the WebLogic Server hosting the OpenSSO STS, to configure SSL, perform the steps described in "Configuring Keystores for SSL" on page 10-36.

    - On Glassfish server hosting the Open SSO STS, perform these steps:

      a. Generate a new key pair for the application server by issuing the following command:

      ```
      keytool -genkey -keyalg <algorithm for generating the key pair>
      -keystore keystore.jks -validity <days> -alias <alias_name>
      ```

      For example:

      ```
      keytool -genkey -keyalg RSA -keystore <glassfish_install_
      dir>/domains/<sts_deploy_domain>/config/keystore.jks -validity
      365 -alias Oracle WSM
      ```

      When prompted for first and last name, enter the hostname of the machine for which the certificate is to be generated. Also enter the appropriate details for the other prompts.

      b. Generate a Certificate Signing Request (CSR) by issuing the following command:

```
keytool -certreq -alias Oracle WSM -file Oracle WSM.csr
-keystore keystore.jks -storepass password
```

The request that is generated and written to the `Oracle WSM.csr` file needs to be submitted to a Certificate Authority in order to get a valid certificate. For example, the Certificate Management Server maintained by the OpenSSO QA team at `https://mahogany.red.iplanet.com`.

c. Access the Certificate Management Server at `https://mahogany.red.iplanet.com`, click **SSL Server** in the left pane, and paste the contents of the `.csr` file, starting from `BEGIN CERTIFICATE REQUEST` and ending at `END CERTIFICATE REQUEST`, into the **PKCS # 10 Request** field.

   Fill out the other fields, as appropriate, and submit the request. Once the request is approved, the certificate can be retrieved from the retrieval tab on the same page.

d. Copy the certificate content (PKCS # 7 format) starting from `BEGIN CERTIFICATE` to `END CERTIFICATE` into a file with `.cert` extension and import the server certificate into the `<glassfish_install_dir>/domains/<sts_deploy_domain>/config/keystore.jks` file by using the following keytool command:

```
keytool -import -v -alias Oracle WSM -file Oracle WSM.cert
-keystore keystore.jks -storepass password
```

   Enter YES when prompted if you trust the certificate.

e. Access the Certificate Authority's SSL Certificate. Go to `https://mahogany.red.iplanet.com` and navigate to **SSL Server -> Retrieval tab -> List Certificates -> Find**. Click on the first **Details** button on the page and copy the Base 64 encoded certificate into another `.cert` file. For example: `mahogany.cert`

f. Import this certificate with alias as "rootca" into the `<glassfish_install_dir>/domains/<sts_deploy_domain>/config/cacerts.jks` file, using the following command:

```
keytool -import -v -alias rootca -file mahogany.cert -keystore
cacerts.jks -storepass password
```

g. The previous step may need to be repeated for client side `truststore.jks` file. Delete any existing `rootca` aliases from that file and import the new one as shown above (changing the location of the keystore file).

h. To configure GlassFish with the new certificate, access the Administration Console at `http://hostname:admin-port/`. Navigate to **Configuration -> HTTP Service -> http-listener2 (default SSL enabled port) -> SSL**, and change the certificate nickname from `s1as` (self-signed cert) to `Oracle WSM`.

i. Restart Glassfish.

10.21.2 SAML Holder-of-Key With Message Protection Scenario

The following procedure describes how to configure SAML holder-of-key with message protection using WS-Trust with OpenSSO STS. This example uses a WebLogic Web service and SOA Composite client to demonstrate the scenario.

To configure SAML holder-of-key with message protection using WS-Trust with OpenSSO STS:

1. Configure OpenSSO STS, as described "Configuring OpenSSO STS" on page 10-111.

2. Configure the STS service policy following the steps described in "Configure a Policy for Automatic Policy Configuration" on page 10-105.

   Make a copy of **oracle/sts_trust_config_service_policy** and edit the policy configuration, as described below, based on the requestor token type.

   To support WS-Security 1.0 username token with message protection requestor token:

   – orasp:port-uri="http://<host>:<port>/openssosts/sts/wss10un"

   – orasp:wsdl-uri="http://<host>:<port>/openssosts/sts/wss10un?wsdl" (Optional)

   To support WS-Security 1.0 username token over SSL with message protection requestor token:

   – orasp:port-uri="https://<host:ssl_port>/openssosts/sts/tlswss10un"

   – orasp:wsdl-uri="https://<host:ssl_port>/openssosts/sts/tlswss10un?wsdl" (Optional)

   To support WS-Security 1.0 X509 token with message protection requestor token:

   – orasp:port-uri="http://<host>:<port>/openssosts/sts/wss10x509"

   – orasp:wsdl-uri="http://<host>:<port>/openssosts/sts/wss10x509?wsdl" (Optional)

   To support WS-Security 1.1 Kerberos token with message protection requestor token:

   – orasp:port-uri="http://<host>:<port>/openssosts/sts/wss11kerberos"

   – orasp:wsdl-uri="http://<host>:<port>/openssosts/sts/wss11kerberos?wsdl" (Optional)

3. Configure the Web service policy following the steps described in "Configure a Web Service for Automatic Policy Configuration" on page 10-107.

   Attach the policy created in step 2 followed by the **oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy** to the WebLogic Web service. For more information, see "Attaching a Policy to a Single Subject" on page 8-3.

   > **Note:** By default, the `oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy` policy is configured with token type of SAML 1.1. If you wish to configure the token type to be SAML 2.0, you will need to make a copy of the policy and edit it, as described in Section 7.5.2, "Creating a Web Service Policy from an Existing Policy". (This value should match the client policy.)

4. Configure the Web service client policy following the steps described in "Configure a Web Service Client for Automatic Policy Configuration" on page 10-106.

   Attach the **oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy** policy to the SOA composite client and override the client configuration properties described in Table 8–4, as required for your requestor token.

The `sts.auth.user.csf.key` should be set to the user credentials available in the default OpenSSO STS configuration. Namely, username and password you specified under the Credential for User Token section for the OpenSSO STS instance. Though, it is not required to be set for the X509 requestor token.

> **Note:** For more information about overriding client configuration properties when attaching a policy, see "Attaching Policies to Web Service Clients" on page 8-11.
>
> By default, the oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy policy is configured with token type of SAML 1.1. If you wish to configure the token type to be SAML 2.0, you will need to make a copy of the policy and edit it, as described in Section 7.5.2, "Creating a Web Service Policy from an Existing Policy". (This value should match the service policy.)

10.21.3 SAML Sender Vouches with Message Protection Scenario

> **Note:** Before proceeding, it is recommended that you review "Using SAML Sender Vouches with WS Trust" on page 10-109.

The following procedure describes how to configure SAML sender vouches with message protection using WS-Trust with OpenSSO STS. This example uses a WebLogic Web service and SOA Composite client to demonstrate the scenario.

To configure SAML sender vouches with message protection using WS-Trust with OpenSSO STS:

1. Configure OpenSSO STS, as described "Configuring OpenSSO STS" on page 10-111.
2. Configure the client-side STS policy following the steps described in "Manually Configuring the STS Config Policy From the Web Service Client: Main Steps" on page 10-107.

   > **Note:** Automatic Policy Configuration cannot be used for SAML sender vouches confirmation because the trust is between the Web service and the client. For more information, see "Using SAML Sender Vouches with WS Trust" on page 10-109.

   Make a copy of `oracle/sts_trust_config_client_policy` and edit the policy configuration based on the requestor token type.

   To support WS-Security 1.0 username token with message protection requestor token:

   – orasp:policy-reference-uri="oracle/wss10_username_token_with_message_protection_client_policy"

   – orasp:port-endpoint="http://<host>:<port>/openfm/SecurityTokenService/#wsdl.endpoint(SecurityTokenService/ISecurityTokenService_Port_UN_WSS10_SOAP12):

   – orasp:port-uri="http://<host>:<port>/openssosts/sts/wss10un"

   – orasp:sts-keystore-recipient-alias="test"

To support WS-Security 1.0 username token over SSL with message protection requestor token:

- orasp:policy-reference-uri="oracle/wss_username_token_over_ssl_client_policy"
- orasp:port-endpoint="http://localhost:8080/openfm/SecurityTokenService/#wsdl.endpoint(SecurityTokenService/ISecurityTokenService_Port_TLS_UN_WSS10_SOAP12)"
- orasp:port-uri="https://<host:ssl_port>/openssosts/sts/tlswss10un"
- orasp:sts-keystore-recipient-alias="test"

To support WS-Security 1.0 X509 token with message protection requestor token:

- orasp:policy-reference-uri="oracle/wss10_x509_token_with_message_protection_client_policy"
- orasp:port-endpoint="http://localhost:8080/openfm/SecurityTokenService/#wsdl.endpoint(SecurityTokenService/ISecurityTokenService_Port_X509_WSS10_SOAP12)"
- orasp:port-uri="http://<host>:<port>/openssosts/sts/wss10x509"
- orasp:sts-keystore-recipient-alias="test"

3. Attach the `oracle/wss11_saml_token_with_message_protection_service_policy` policy to the WebLogic Web service (there is no corresponding issued token policy for SAML sender vouches scenarios) and override the `keystore.enc.csf.key` to specify the service encryption key alias and password.

   > **Note:** By default, the oracle/wss11_saml_hok_with_message_protection_service_policy policy is configured with token type of SAML 1.1. If you wish to configure the token type to be SAML 2.0, you will need to make a copy of the policy and edit it, as described in Section 7.5.2, "Creating a Web Service Policy from an Existing Policy".

4. Attach the policy created in step 2 followed by the `oracle/wss11_sts_issued_saml_with_message_protection_client_policy` policy to the SOA composite client and override the client configuration properties described in Table 8–4, as required for your requestor token.

   The "On Behalf Of" use case relies on the `sts.auth.on.behalf.of.csf.key` and `on.behalf.of` properties described in Table 8–4. For more information, see "On Behalf Of Use Cases" on page 10-100.

   The `on.behalf.of` property should be set to `true`. The `sts.auth.on.behalf.of.csf.key` should be set to the user credentials available in the default Open SSO STS configuration that support the "on behalf of" use case. Namely, `demo`, with password set to *`password`*.

   > **Note:** For more information about overriding client configuration properties when attaching a policy, see "Attaching Policies to Web Service Clients" on page 8-11.

5. To grant permission to the client application to request a token from OpenSSO STS "on behalf of" a user, edit the `<MW_HOME>/user_projects/domains/base_`

`domain/config/fmwconfig/system-jazn-data.xml` file to include the following code:

```
<grant>
   <grantee>
      <codesource>
         <url>
file:${common.components.home}/modules/oracle.wsm.agent.common_
${jrf.version}/wsm-agent-core.jar
         </url>
      </codesource>
   </grantee>
   <permissions>
      <permission>
            <class>oracle.wsm.security.WSIdentityPermission</class>
            <name>resource=<Client App. Name></name>
            <actions>assert</actions>
      </permission>
    </permissions>
</grant>
```

10.21.4 SAML Bearer with Message Protection Scenario

The following procedure describes how to configure SAML bearer with message protection using WS-Trust with OpenSSO STS. This example uses a WebLogic Web service and SOA Composite client to demonstrate the scenario.

To configure SAML bearer with message protection using WS-Trust with OpenSSO STS:

1. Configure OpenSSO STS. as described "Configuring OpenSSO STS" on page 10-111.
2. Configure the STS policy following the steps described in "Configure a Policy for Automatic Policy Configuration" on page 10-105.

   Make a copy of `oracle/sts_trust_config_service_policy` and edit the policy configuration, as described below, based on the requestor token type.

   To support WS-Security 1.0 username token with message protection requestor token:

   – orasp:port-uri="http://<host>:<port>/openssosts/sts/wss10un"

   – orasp:wsdl-uri="http://<host>:<port>/openssosts/sts/wss10un?wsdl" (Optional)

   To support WS-Security 1.0 username token over SSL with message protection requestor token:

   – orasp:port-uri="https://<host:ssl_port>/openssosts/sts/tlswss10un"

   – orasp:wsdl-uri="https://<host:ssl_port>/openssosts/sts/tlswss10un?wsdl" (Optional)

   To support WS-Security 1.0 X509 token with message protection requestor token:

   – orasp:port-uri="http://<host>:<port>/openssosts/sts/wss10x509"

   – orasp:wsdl-uri="http://<host>:<port>/openssosts/sts/wss10x509?wsdl" (Optional)

To support WS-Security 1.1 Kerberos token with message protection requestor token:

– orasp:port-uri="http://<host>:<port>/openssosts/sts/wss11kerberos"

– orasp:wsdl-uri="http://<host>:<port>/openssosts/sts/wss11kerberos?wsdl" (Optional)

3. Configure the Web service policy following the steps described in "Configure a Web Service for Automatic Policy Configuration" on page 10-107.

   Attach the policy created in step 2 followed by the `oracle/wss11_sts_issued_saml_bearer_token_over_ssl_service_policy`. For more information, see "Attaching a Policy to a Single Subject" on page 8-3.

4. Configure the Web service client policy following the steps described in "Configure a Web Service Client for Automatic Policy Configuration" on page 10-106.

   Attach the `oracle/ws11_sts_issued_saml_bearer_token_over_ssl_client_policy` policy to the SOA composite client and override the client configuration properties described in Table 8–4, as required for your requestor token.

   The `sts.auth.user.csf.key` should be set to the user credentials available in the default OpenSSO STS configuration. Namely, username and password Under the Credential for User Token section for the OpenSSO STS instance. Though, it is not required to be set for the X509 requestor token.

> **Note:** For more information about overriding client configuration properties when attaching a policy, see "Attaching Policies to Web Service Clients" on page 8-11.

10.22 Understanding Fine-Grained Authorization Using Oracle Entitlements Server

Oracle Entitlements Server (OES) is a fine-grained authorization service you can use to secure applications and services across the enterprise. It supports centralized definition of complex application entitlements and the distributed runtime enforcement of those entitlements. OES allows you to externalize entitlements and thereby remove security decisions from the application.

Oracle WSM OES integration supports advanced authorization use cases using OES and provides the following capabilities:

- You can apply OES authorization to your SOAP-based web services. The OES authorization policy provides a grant or deny for a subject to perform a certain action on a given resource.
- OES can make grant/deny decisions based on context attributes. The context attributes could be based on information from the SOAP request message extracted using XPath statements, or they could be based on HTTP headers.
- Data masking. Oracle WSM OES can mask (with character of your choice) certain information in the response for the web service request.

This section describes how Oracle Entitlements Server (OES) is integrated with Oracle WSM, and how you can use OES together with Oracle WSM for fine-grained authorization.

The following topics are described:

- "Prerequisite OES Reading" on page 10-119
- "OES Integration: The Big Picture" on page 10-119
- "Oracle WSM OES Policies" on page 10-124
- "Resource Mapping and Naming" on page 10-125
- "How Attributes Are Processed" on page 10-127
- "Use of Guard Element" on page 10-130

See "Configuring Fine-Grained Authorization Using Oracle Entitlements Server" on page 11-148 for configuration information.

10.22.1 Prerequisite OES Reading

This section references many OES concepts and features. However, the focus of the section is the integration with Oracle WSM, and it does not attempt to provide an in-depth discussion of the OES concepts.   If you are not already familiar with OES, you should first refer to the following OES documentation:

> **Note:** Oracle WSM supports version 11.1.2.2.0 or later of OES.

- *Administrator's Guide for Oracle Entitlements Server*
- Fine Grained Authorization: Technical Insights for using Oracle Entitlements Server

10.22.2 OES Integration: The Big Picture

When you integrate Oracle WSM and OES, you:

- Attach an Oracle WSM authentication policy to your web service.
- Attach the Oracle WSM `oracle/binding_oes_authorization_policy` or `oracle/component_oes_authorization_policy` policy, alone or in combination with the `oracle/binding_oes_masking_policy` policy as described in this section.
- Use the OES console to create authorization and data masking policies, typically with separate policies for Obligations.

> **Note:** Oracle WSM does not expose any OES-related configuration; you use the OES console for this purpose.

The Oracle WSM agent checks the authorization of a soap request for a protected web service based on the policies defined in OES.   To do this, Oracle WSM passes to OES the authenticated subject, the target resource and requested action, as well as a set of implicit attributes that are always passed in authorization requests.

In your OES policy you can define additional required values based on context attributes from the SOAP request, HTTP headers, message context properties or identity information like the subject, roles, and groups. If you configure OES to require any extra of these context attributes to make a permit/deny decision, Oracle WSM passes them as well.

Specifically, there are two ways to contact OES for the authorization decision: a two-step method and a single-step method.   You select which via the use.single.step attribute in oracle/binding_oes_authorization_policy and oracle/component_oes_authorization_policy.

The methods function as follows:

- In the two-step process, you must have previously identified attributes required for fine-grained authorization in the OES console and you now want Oracle WSM to use them.

  Oracle WSM first calls to OES to find out what attributes are needed, gets the attributes from the request payload, and then calls OES a second time to perform the actual authorization using the OES authorization policy. This means that you actually define two OES policies: one to get the needed attributes, and one for the authorization itself.

  You can also use always-passed implicit attributes, plus OES predefined attributes such as time, date, and so forth.

  This method is used for **fine-grained authorization**, as described in "OES Fine- and Coarse-Grained Authorization" on page 10-121.

- In the single-step process, Oracle WSM makes only one call to OES to perform the authorization using the OES authorization policy. The single-step process does not require any previously-identified attributes. As with the two-step process, you can also use the always-passed implicit attributes, plus OES predefined attributes such as time, date, and so forth.

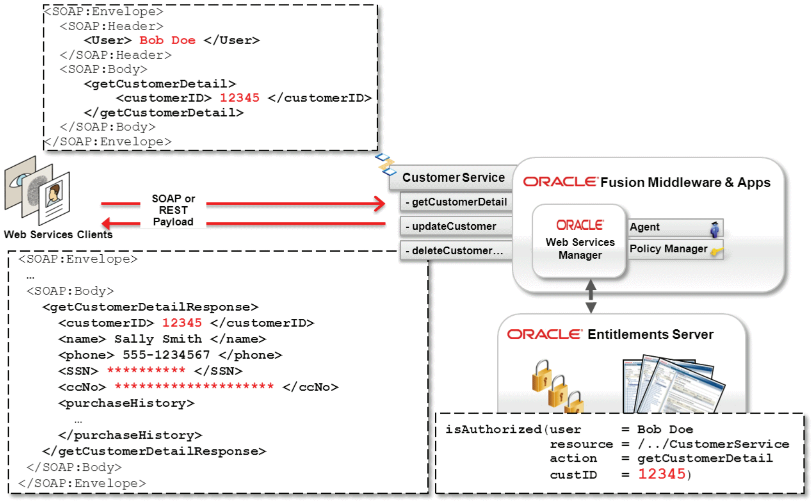
  This method can be used for **coarse-grained authorization**, as described in "OES Fine- and Coarse-Grained Authorization" on page 10-121.

10.22.2.1 Data Masking

Oracle WSM with OES integration can mask (with asterisks) certain information in the response from the web service, without changing any of the web services code.

Assume you want to ensure that sensitive data is not passed over the wire in response to a web service client request. You use the OES console together with the oracle/binding_oes_masking_policy policy to replace any sensitive data leaving the web service such as a social security number or financial information with asterisks, as shown in Figure 10–31.

Figure 10–31 Masking Sensitive Data



```
<SOAP:Envelope>
  <SOAP:Header>
     <User> Bob Doe </User>
  </SOAP:Header>
  <SOAP:Body>
     <getCustomerDetail>
         <customerID> 12345 </customerID>
     </getCustomerDetail>
  </SOAP:Body>
</SOAP:Envelope>
```

```
<SOAP:Envelope>
 …
 <SOAP:Body>
   <getCustomerDetailResponse>
     <customerID> 12345 </customerID>
     <name> Sally Smith </name>
     <phone> 555-1234567 </phone>
     <SSN> ********** </SSN>
     <ccNo> ******************** </ccNo>
     <purchaseHistory>
       …
     </purchaseHistory>
   </getCustomerDetailResponse>
 </SOAP:Body>
</SOAP:Envelope>
```

```
isAuthorized(user     = Bob Doe
             resource = /../CustomerService
             action   = getCustomerDetail
             custID   = 12345)
```

Masking sensitive data is based on who asked for it, and on other context attributes present in the request.

Consider the following code flow for the web service response shown in Figure 10–31.

Data Masking Code Flow

1. The web service client sends a request.
2. On the inbound request, Oracle WSM enforces the request policy and performs the appropriate authentication and authorization for user `Bob Doe`.
3. If the request is permitted, Oracle WSM passes the payload to the service provider. The service provider acts on the payload and prepares a response to be sent back to the caller.
4. During response processing, Oracle WSM invokes the `oracle/binding_oes_masking_policy` policy to determine if there is any sensitive data that needs to be masked.

   Oracle WSM passes the caller's information and any of the user-defined attributes extracted from the response payload.
5. The data masking rules defined in OES take into consideration the client information (through transport attributes), the current subject, resource, action and any response attributes configured on the policy.
6. For each payload attribute, OES responds with Obligations that specify whether the attribute should be passed as-is, or masked.
7. Oracle WSM honors the Obligations returned by OES and masks attributes marked as sensitive by OES.

10.22.2.2 Obligations

OES supports the XACML concept of **Obligations**. As described in "Understanding the Policy Model" in *Administrator's Guide for Oracle Entitlements Server*, when used in a policy, an Obligation may impose an additional requirement for the policy enforcing component.

You configure the Obligation in the OES console. An Obligation is any attribute name/value pair (or any other simple name/value pair) that is returned back to the caller (Oracle WSM). For Oracle WSM OES integration, the Obligation can be an XPath query, HTTP transport header properties, or message context properties.

Another use of Obligations is data masking. In certain applications, such as data security use-cases, a simple yes or no answer may not be sufficient and the OES authorization policy might return an Obligation that specifies what data is to be masked and with what value, as previously shown in Figure 10–31.

10.22.2.3 OES Fine- and Coarse-Grained Authorization

There are two ways to do authorization with the Oracle WSM OES policies: fine- and coarse-grained authorization. The authorization types are defined as follows:

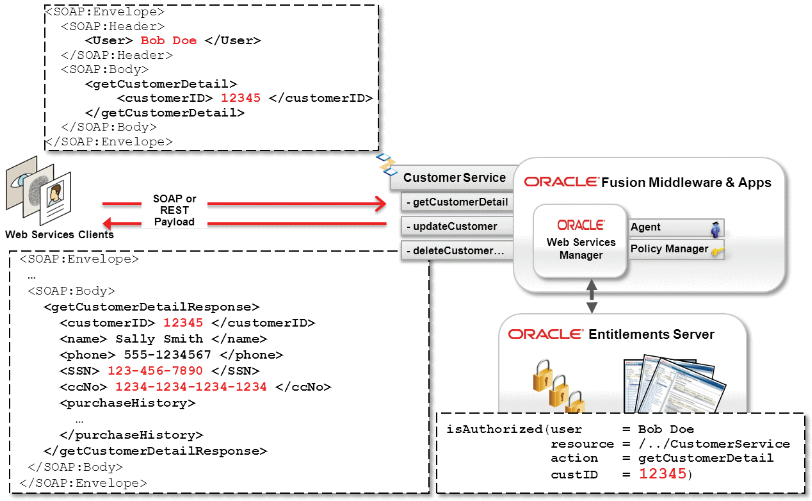
- Fine (Obligations) — You want to determine access to the resource based on the identity of the consumer, plus specific content from the transport header or the payload specified in Obligations. This is the common use case.

  In this use case, you define attributes in the OES access policy that Oracle WSM will then extract from the request and pass back to OES during authorization. That is, the OES access policy is based on a combination of identity attributes or attributes extracted from the request payload.

For example, you might have an OES access check of "Allow access if the SOAP Body contains a particular customer ID and if the authenticated user belongs to group TrustedPartners."

Figure 10–32 shows the fine-grained authorization use case.

Figure 10–32 Fine-Grained Authorization



```
<SOAP:Envelope>
  <SOAP:Header>
     <User> Bob Doe </User>
  </SOAP:Header>
  <SOAP:Body>
     <getCustomerDetail>
         <customerID> 12345 </customerID>
     </getCustomerDetail>
  </SOAP:Body>
</SOAP:Envelope>
```

```
<SOAP:Envelope>
  …
  <SOAP:Body>
    <getCustomerDetailResponse>
      <customerID> 12345 </customerID>
      <name> Sally Smith </name>
      <phone> 555-1234567 </phone>
      <SSN> 123-456-7890 </SSN>
      <ccNo> 1234-1234-1234-1234 </ccNo>
      <purchaseHistory>
        …
      </purchaseHistory>
    </getCustomerDetailResponse>
  </SOAP:Body>
</SOAP:Envelope>
```

```
isAuthorized(user     = Bob Doe
             resource = /../CustomerService
             action   = getCustomerDetail
             custID   = 12345)
```

Consider the following code flow for the web service response shown in Figure 10–32.

1. The web service client sends a SOAP request.
2. The web service is secured with an Oracle WSM authentication policy and an Oracle WSM OES authorization policy.
3. Oracle WSM performs authentication and invokes OES for authorization.
4. Oracle WSM provides the subject, resource, lookup action and all predefined properties to OES.
5. OES calls the lookup action configured for the protected resource and responds with the configured Obligations (if any) to Oracle WSM. Returned obligations can be returned based on actions; for each action you can define different XPaths, and so forth.
6. Oracle WSM evaluates the Obligations and executes the XPath on the SOAP/XML payload or finds the property values from the transport header or message context.
7. Oracle WSM again provides the subject, resource, and action, plus all of the attributes evaluated in Step 6 to OES.
8. OES determines access based on the subject, resource, action and attributes.
9. OES responds with permit or deny.
10. If permit, Oracle WSM passes on the message to the service provider.
11. If deny, Oracle WSM rejects the request with an authentication failure fault.

- Fine-grained with SAML — You want to determine access to the resource based on the identity of the consumer and on the attributes passed in a SAML token.

Oracle WSM passes attributes from a SAML assertion. SAML attributes are part of the implicit attributes that are always extracted (if present) and sent automatically. The name of the attribute is the name of the attribute inside the SAML assertion and the value is the list of strings.   OES can determine access based on these SAML attributes, as well as the subject, resource and action.

There are two ways to implement this use case. The first approach is to create an OES custom attribute retriever, as described in "Creating Custom Attribute Retrievers" in *Developer's Guide for Oracle Entitlements Server*. The second approach is to have OES respond using Obligations with XPaths that point to the SAML attribute values.

Consider the following code flow for the approach of using an OES custom attribute retriever:

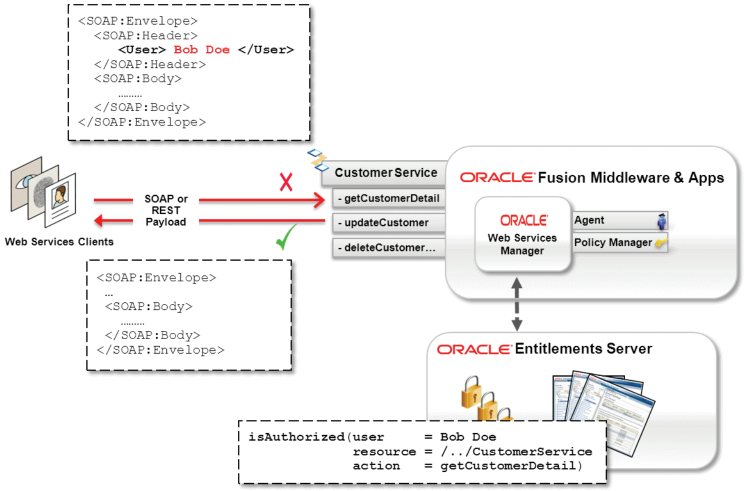
1. The web service client sends a SAML token with an attribute statement in a SOAP request.
2. The web service is secured with `oracle/wss10_saml_token_service_policy` and an Oracle WSM OES authorization policy.
3. Oracle WSM performs authentication and checks the SAML assertion for an `AttributeStatement`. If attributes are present, then Oracle WSM extracts them and passes them as attributes while invoking OES for the access request.
4. Oracle WSM provides the subject, resource, and action, along with any other pre-defined attributes.
5. OES determines access based on the SAML attributes, subject, resource and action. OES uses a custom attribute retriever to get the SAML attributes.
6. OES responds with permit or deny.
7. If permit, Oracle WSM passes on the message to the service provider.
8. If deny, Oracle WSM rejects the request with an access-denied fault.

- Coarse — You want OES to determine access to the resource based on the identity of the consumer and the web service operation being called. The OES access check is based on the identity attributes, which are limited to user name, group, and role. You can also use the implicit attributes, plus OES predefined attributes such as time, date, and so forth.

  You must set `use.single.step` to `true` in the Oracle WSM OES policy to use this mode.

  Figure 10–33 shows the coarse-grained authorization use case. In this use case, assume that you want to secure the service with an authorization policy that determines whether the consumer is allowed to access the service. You want to determine access to the resource based on the identity (authenticated subject) of the consumer and the web service operation being invoked. For example, in Figure 10–33 user `Bob Doe` might be authorized to get the customer detail but not to delete the customer record.

Figure 10–33 Coarse-Grained Authorization


```
<SOAP:Envelope>
  <SOAP:Header>
     <User> Bob Doe </User>
  </SOAP:Header>
  <SOAP:Body>
    .........
  </SOAP:Body>
</SOAP:Envelope>
```

```
<SOAP:Envelope>
 ...
 <SOAP:Body>
   .........
 </SOAP:Body>
</SOAP:Envelope>
```

```
isAuthorized(user     = Bob Doe
             resource = /../CustomerService
             action   = getCustomerDetail)
```

Consider the following code flow for the web service response shown in Figure 10–33.

1. The web service client sends a SOAP or XML request.
2. The web service is secured with an Oracle WSM authentication policy and an Oracle WSM OES policy.
3. Oracle WSM performs authentication and invokes OES for the access request. Oracle WSM provides the subject, resource and action information to OES.
4. OES determines access based on the subject, resource and action information.
5. OES responds with permit or deny.
6. If permit, Oracle WSM passes on the message to the service provider.
7. If deny, Oracle WSM rejects the request with an access-denied fault.

10.22.3 Oracle WSM OES Policies

Oracle WSM includes the following OES authorization and masking policies:

- `oracle/binding_oes_authorization_policy` — This policy does user authorization based on the policy defined in OES. Authorization is based on attributes, the current authenticated subject, and the web service action invoked by the client.

  This policy is used for coarse- or fine-grained authorization on any operation on a web service, as determined by the `use.single.step` attribute. (See "OES Fine- and Coarse-Grained Authorization" on page 10-121.)

  You must use an authentication policy with the Oracle WSM OES authorization policy because the Oracle WSM OES policy requires an authenticated subject.

  This policy also uses the guard element (see orawsp:guard)   to define resource, action, and constraint match values.   These values allow the assertion execution only if the result of the guard is true.   If the accessed resource name and action match, only then is the assertion allowed to execute. By default, resource name and action use the wildcard asterisk "*" and everything is allowed.

  This policy can be attached to any SOAP-based endpoint.

- `oracle/component_oes_authorization_policy` — This policy does user authorization based on the policy defined in OES.

  This policy is used for coarse- or fine-grained authorization on any operation on a SOA component, as determined by the `use.single.step` attribute. (See "OES Fine- and Coarse-Grained Authorization" on page 10-121.)

  You must use an authentication policy with the Oracle WSM OES authorization policy because the Oracle WSM OES policy requires an authenticated subject. Authorization is based on attributes, the current authenticated subject, and the web service action invoked by the client.

  This policy also uses the guard element (see orawsp:guard) to define resource, action, and constraint match values. These values allow the assertion execution only if the result of the guard is true. If the accessed resource name and action match, only then is the assertion allowed to execute. By default, resource name and action use the wildcard asterisk "*" and everything is allowed.

  This policy is used for fine-grained authorization on a SOA component.

- `oracle/binding_oes_masking_policy` — This policy does response masking based on the policy defined in OES. You can use an authentication policy with the Oracle WSM OES masking policy. (If there is no subject, the masking decision does not consider the user when making a decision.) Masking is based on attributes, the current authenticated subject, and the web service action invoked by the client.

  This policy uses the guard element (see orawsp:guard) to define resource, action, and constraint match values. These values allow the assertion execution only if the result of the guard is true. If the accessed resource name and action match, only then is the assertion allowed to execute. By default, resource name and action use the wildcard asterisk "*" and everything is allowed.

  This policy is used for fine-grained masking on any operation of a web service.

10.22.4 Resource Mapping and Naming

You must map the OES resource name to the Oracle WSM resource name. When making an authorization call from Oracle WSM, the resource name is passed to OES, and this name must exactly match the one defined in the OES policy.

Table 10–7 shows how to construct the resource string for the OES policy.

If you follow the naming conventions, you do not have to set the resource name in the Oracle WSM policy, Oracle WSM derives it.

> **Note:** This is the default mapping. If you need to change this mapping, use configuration overrides, as described in "Configuration Properties and Overrides" on page 11-161.

Table 10–7 Determining Resource String

OES Field	Value to Use
Application	Deployed Application Name. For SOA, the composite name is used as the application name.

Table 10–7 (Cont.) Determining Resource String

OES Field	Value to Use
Resource Type	Fixed, based on subject type. - For SOAP must be `WS_SERVICE`. - For SOA component, must be `COMPONENT`.
Resource Name	- For SOAP and SOA reference, must be of the form `web-service-name/port/web service operation`. - For SOA component, must be of the form `SOA component name/web service operation`.
Action	By default, one of: `request.lookup` (Obligation policy for authorization.) `response.lookup` (Obligation policy for masking.) `mask` (Real masking policy.) `authorize` (Real authorization policy.)

10.22.4.1 Example of OES Policies

Assume that a SOA composite (soa1) has two service bindings (`Serv1` and `Serv2`).

- `Serv1` has `port11`
- `Serv2` has `port21`
- `port11` has `oper11`, `oper12`
- `port21` has `oper21`

In OES, the application, resource type, resource name and actions should be defined as shown in Table 10–8.

Table 10–8 Resource String Example

OES Field	Value to Use
Application	soa1
Resource Type	`WS_SERVICE`

Resource Name	`Serv1/port11/oper11,`
	`Serv1/port11/oper12,`
	`Serv2/port21/oper21`

Action	One of: `request.lookup` (Obligation policy for authorization.) `response.lookup` (Obligation policy for masking.) `mask` (Real masking policy.) `authorize` (Real authorization policy.)

The authorization and masking OES policies based on Table 10–8 are as follows:

- Returning Obligations
  - One policy that returns obligations for any operation:

    ```
    GRANT (action: request.lookup; Resource: WS_SERVICE/Serv1/Port11, WS_
    SERVICE/Serv2/Port21; User: any) Obligation: XPath11
    ```
  - Multiple policies for returning operation-specific Obligations:

    ```
    GRANT (action: request.lookup; Resource: WS_
    SERVICE/Serv1/Port11/oper11;User:any) Obligation: XPath11
    GRANT (action: request.lookup; Resource: WS_
    SERVICE/Serv1/Port11/oper12;User:any) Obligation: XPath12
    GRANT (action: request.lookup; Resource: WS_
    SERVICE/Serv2/Port21/oper21;User:any) Obligation: XPath21
    ```
- Real authorization
  - One policy performing same authorization regardless of resource and action:

    ```
    GRANT/DENY (action: authorize; Resource: WS_SERVICE/Serv1/Port11, WS_
    SERVICE/Serv2/Port21; User:<actual user>)
    ```
  - Multiple policies for performing operation-specific authorization:

    ```
    GRANT/DENY (action: authorize; Resource: WS_
    SERVICE/Serv1/Port11/oper11;User:<actual user>)
    GRANT/DENY (action: authorize; Resource: WS_
    SERVICE/Serv1/Port11/oper12;User:<actual user>)
    ```
- Returning masking Obligations:

  ```
  GRANT (action: response.lookup; Resource: WS_
  SERVICE/Serv1/Port11/oper11;User:any) Obligation: XPath11
  GRANT (action: response.lookup; Resource: WS_
  SERVICE/Serv1/Port11/oper12;User:any) Obligation: XPath12
  GRANT (action: response.lookup; Resource: WS_
  SERVICE/Serv2/Port21/oper21;User:any) Obligation: XPath21
  ```
- Real masking:

  ```
  GRANT/DENY (action: mask; Resource: WS_SERVICE/Serv1/Port11/oper11;User:<actual
  user>)
  GRANT/DENY (action: mask; Resource: WS_SERVICE/Serv2/Port21/oper21;User:<actual
  user>)
  ```

10.22.5 How Attributes Are Processed

As the OES administrator, you define attributes in the OES policy as Obligations, which Oracle WSM then extracts from the payload and sends back to OES.

Specifically, OES allows you to create an Obligation in the OES console and provide multiple attribute name/value pairs. For example, you can create an Obligation called `Employee` and have multiple attributes such as `{Name=John, Age=21, SSN=123456}`.

The attributes can be obtained from an XPath, an HTTP header, a message context, and constants (name/value). These attributes must follow a specific naming convention, as described in Table 10–9.

Table 10–9 Attribute Types Supported for OES Policies

Attribute Type	Description	Required Format

XPath query	You provide the attribute name and value as an XPath query in the OES console.
	Oracle WSM runs this XPath query on the SOAP message and uses the value as the attribute value. The XPath query can result in a single value or multiple values. In case of multiple values, a list of strings is used to pass all values.
	If any XPath query fails to evaluate on the SOAP message, it is ignored and a warning message is generated in the logs. Oracle WSM continues to evaluate next XPath query.
		Use `XPath` (case insensitive) as the Obligation name to signify that it is an XPath.
		The Obligation should also return all the namespaces being used in the XPath query. All namespaces should be returned with an attribute name of `NAMESPACE` (case insensitive) and the value should be the comma separated namespaces.
		For example, if you want to use the SAML issuer name for authorization, use the following Obligation format:
		`Name = XPath, values = {saml_issuer=.//saml:Assertion/@Issuer, CC_Name=ns1:sayHello/arg0, NAMESPACE=ns1=http://...,wsse=http://...}`
		In the authorization phase, Oracle WSM passes the attribute name `saml_issuer` and the value is the result of the XPath query. The default namespace has to be mapped to a prefix. (The prefix name must be unique within the application.)
		For example:
		`saml=urn:oasis:names:tc:SAML:1.0:assertion,ns0=http://wsm.oracle.com,myPrefix=http://default_namespace`
		Namespace definitions are separated using a comma.

HTTP Header	You provide HTTP header names in the OES console. The value is fetched from the current request HTTP header.	To get HTTP Header properties, define an Obligation with the name "HTTPHeader" (case insensitive). It can have multiple HTTP header names. The name of the attribute should be the name to which you want to assign the value; the value should be the actual HTTP header name. For example: `Name = HTTPHeader, values = {AuthHeader=Authorization}` In the authorization phase, Oracle WSM retrieves the HTTP header and assigns it to the name given in the attribute name.

Table 10–9 (Cont.) Attribute Types Supported for OES Policies

Attribute Type	Description	Required Format

Message Context Properties	You provide message context property names in the OES console.
	The value is fetched from the current message context.
		Define an Obligation with the name "MessageContext" (case insensitive). It can have multiple message context property names.
		The name of the attribute should be the name to which you want to assign a value; the value should be the actual message context property name.
		For example:
		`Name = MessageContext, values = {authMethod=oracle.wsm.internal.authentication.method, endpoint=oracle.j2ee.ws.runtime.endpoint-url}`
		In the authorization phase, Oracle WSM retrieves the message context property and assigns it to the name given in the attribute name.
		For example, the previous example might resolve to:
		`authMethod=USERNAME_TOKEN & endpoint=http://localhost:7001/myService`

Constants	Constants are user-defined attributes that Oracle WSM does not understand and passes "as is."	An Obligation named `Employee` is an example of a constant.

Table 10–9 (Cont.) Attribute Types Supported for OES Policies

Attribute Type	Description	Required Format
Implicit	Oracle WSM passes implicit attributes in all authorization requests. You do not perform any configuration to pass them. The following implicit attributes are always passed: ■ `serviceURL` — The URL of the web service. ■ `serviceNS` — The namespace of the web service. ■ `clientIP` — The client's IP address. ■ `processingStage` — Whether this is a request or response. Possible values are request, response, and fault. ■ `isRequestOverSSL` — Boolean. True if the request is over one- or two-way SSL.) ■ `authenticationMethod` — The authentication method. Possible values are `SAML_SV, KERBEROS`, `SAML_HOK`, `X509_TOKEN_AUTHENTICATION`, `SAML_BEARER`, and `USERNAME_TOKEN` ■ `requestOrigin` — Where the request came from, internal or external, as determined from the VIRTUAL_HOST_TYPE transport header. ■ `clientSigningCertDN` — Either the X509 signing cert or the client cert in two-way SSL. ■ `operationName` — The operation name invoked by the user. ■ `samlIssuer` — The SAML issuer extracted from the SAML assertion. ■ `type` — The type of the request to OES. Values can be request.lookup, response.lookup, authorize or mask. This attribute is always sent.	None required, they are always passed. You would typically use these constants in a Condition in the OES console.

10.22.6 Use of Guard Element

The Oracle WSM OES authorization policies uses the orawsp:guard element. It allows the assertion to execute only if the result of the guard is true. That is, if the accessed resource name and action match, only then is the OES authorization engine called.

By default, resource name and action use the wildcard asterisk "*" and everything is allowed. However, if you set a specific resource name, action, and constraint, that requirement must be satisfied before any of the configuration properties and any OES policies are considered.

The resource naming convention for guard differs from the OES standard naming convention. The resource name for the guard must be in the form `<Webservice_NS>/<SERVICE_NAME>`.

11 Configuring Policies

This chapter discusses how to configure policies in Web services and Web service clients to achieve Quality of Service (QoS) requirements.

The predefined policies are described in Appendix B, "Predefined Policies". This Appendix is the definitive source of information for the format of the policies. Some information from the Appendix is repeated here for your convenience.

This chapter includes the following sections:

11.1 Determining Which Security Policies to Use

Note: To secure servlet applications, such as ADF business components exposed as RESTful servlets, you can attach one or more of the HTTP-based authentication or authorization policies listed in Table 8–1.

Use the following series of questions to help you identify the security policies that best meet your requirements:

1. What are the **basic requirements** of your security policy? Decide if you need to only authenticate users, or if you only need message protection, or if you need both.

    a. Do you require authentication only? If yes, then go to step 2.

    b. Do you require authorization only? If yes, then see "Authorization Policies and Configuration Steps" on page 11-98.

    c. Do you require authentication and authorization? If yes, then go to step 3.

    d. Do you only require message protection? If yes, then see "Message Protection-Only Policies and Configuration Steps" on page 11-28.

    e. Do you require both authentication and message protection? If yes, then go to step 4.

2. If you only require **authentication**, then there are two basic questions you need to consider:

    a. Where will the token be inserted? Will the token to be inserted in the transport layer or in a SOAP header?

    b. Do you need to use a particular type of token? The supported credentials for authentication-only policies are username/password, SAML, JWT, and Kerberos tokens.

3. If you require **authentication and authorization**, then you need to consider the following:

    a. Review the considerations provided for authentication in step 2.

    b. Review "Authorization Policies and Configuration Steps" on page 11-98 for more information about authorization policies.

4. If you require both **authentication and message protection**, then you need to consider the following:

    a. Will message protection be handled in the transport layer? If yes, then there are four sets of policies to choose from: Username over SSL, SAML over SSL (Sender-Vouches), SAML over SSL (Token Bearer), and HTTP token over SSL.

        In one set of policies (wss_http_token_over_ssl_client_policy and wss_http_token_over_ssl_service_policy) authentication is also handled in the transport layer. For the other three polices, authentication takes place in the SOAP header.

        If you are using the WS-Security V1.0 or V1.1 standard, then both authentication and message protection occur in the SOAP header. There are five pairs of policies supporting the following tokens: username/password, SAML, and X.509 certificates.

        For more information, see "Message Protection and Authentication Policies and Configuration Steps" on page 11-34.

11.2 Protecting Messages

Message protection involves encrypting the message for message confidentiality and signing the message for message integrity. Oracle Fusion Middleware predefined policies and any policy you create using one of the message-protection assertion templates provide the options for message confidentiality, message integrity, or both.

The following steps summarizes what you must do to configure the clients and services for message protection:

- Attach the appropriate message protection policy to each of the clients and services.
- If you want message integrity, then the message must be signed.
- If you want message confidentiality, then the message must be encrypted.
- Add the required public and private keys to the keystores of the clients and services. This step requires you to configure the keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

11.2.1 Message Protection Basics

Message protection encompasses two concepts, **message confidentiality** and **message integrity**.

Message confidentiality involves keeping the data secret, as well as the identities of the sending and receiving parties. Confidentiality is achieved by encrypting the content of messages and obfuscating the identities of the sending and receiving parties.   The sender uses the recipient's public key to encrypt the message. Only the recipient's private key can successfully decrypt the message, ensuring that it cannot be read by third parties while in transit.   The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

Message integrity is achieved by having an authority digitally sign the message. Digital signatures are used to authenticate the sender of the SOAP message and to ensure the integrity of the SOAP message (that is, to ensure that the SOAP message is not altered while in transit).

When a digital signature is applied to a SOAP message, a unique hash is produced from the message, and this hash is then encrypted with the sender's private key. When the message is received, the recipient decrypts the hash using the sender's public key.

> **Note:** Generally, the recipient does not need to have the sender's public key in its keystore to validate the certificate. It is sufficient to have the root certificate in the keystore to verify the certificate chain. However, if the sender's public key is not present in the message, as in the case of the Thumbprint and SerialIssuer mechanisms, the sender's public key must be in the recipient's keystore.

This serves to authenticate the sender, because only the sender could have encrypted the hash with the private key. It also serves to ensure that the SOAP message has not been tampered with while in transit, because the recipient can compare the hash sent with the message with a hash produced on the recipient's end.

The message-protection assertion templates and predefined policies can be used to protect request and response messages by doing the following:

- Signing messages
- Encrypting messages
- Signing *and* encrypting messages
- Decrypting messages

- Verifying signatures
- Decrypting messages *and* verifying signatures

The Fusion Middleware Control user interface for the predefined message protection policies makes it easy to specify which message parts are signed, encrypted, or both. You can require that the entire body be signed, encrypted, or both, or identity specific header and body elements. The following is an example of partial encryption.

11.2.1.1 Example for Partial Encryption

In this example, a part of the SOAP message is encrypted using Fusion Middleware Control:

1. Create a simple Web service that approves a credit card number (cardNr). A sample payload is shown in Example 11–1.

Example 11–1 Example of a Payload

```
<soapenv:Body     wsu:Id="Body-2grW1pYwjwsoskbLuMJZzg22"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-ws
security-utility-1.0.xsd">


    <aaav:validateTheCard     xmlns:aaav="http://aaavalidatecred/">
      <aaav:cardNr>string</aaav:cardNr>
       <aaav:firstName>string</aaav:firstName>
       <aaav:lastName>string</aaav:lastName>
       <aaav:validUntilDate>string</aaav:validUntilDate>
    </aaav:validateTheCard>

   </soapenv:Body>
```

2. In Fusion Middleware Control, select a message protection policy and click Edit.
3. In the Settings tab, select the Request tab.
4. In the Message Encrypt Setting section, deselect Include Entire Body (Figure 11–1).
5. Expand Body Elements and click Add.
6. Enter the Namespace and the Element Name. In this example, only the card number is encrypted as follows:

   Namespace = `http://aaavalidatecred/`

   Element Name = `cardNr`

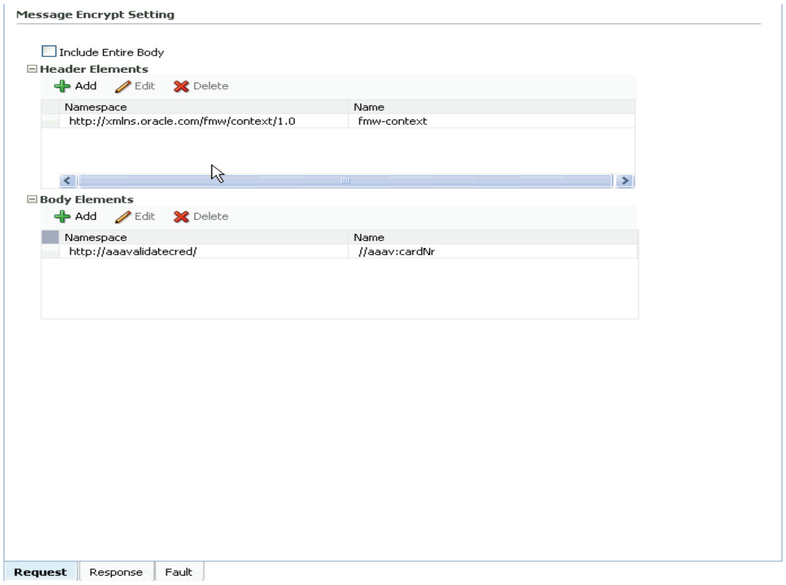
   For more information on other fields in the Edit Policy page, see Table C–118. Example 11–2 shows what the policy would look like.

Example 11–2 Sample Policy with Partial Encryption

```
 <orasp:encrypted-elements>
              <orasp:element orasp:namespace="http://aaavalidatecred/"
orasp:name="cardNr">n/a</orasp:element>
   </orasp:encrypted-elements>
```

7. Click Yes to add the Body Elements and Save to save the modified policy.

Figure 11–1 Example of Partial Encryption of Message Protection Policies



Message Encrypt Setting

□ Include Entire Body

Header Elements

Namespace	Name
http://xmlns.oracle.com/fmw/context/1.0	fmw-context

Body Elements

Namespace	Name
http://aaavalidatecred/	//aaav:cardNr

11.2.1.2 Security SwA Attachments

Packaging as attachments in SOAP messages has become common for any data that cannot be placed inside SOAP Envelope. The primary SOAP message can reference additional entities as attachments or attachments with MIME headers.

Each SwA attachment is a MIME part and contains the MIME header. **Include SwA Attachment** signs the attachment but not the MIME header corresponding to that. **Include MIME Headers** signs the corresponding MIME headers as well as the attachments.

11.2.2 Which Policies Offer Message Protection?

The following policies offer message protection. The subsequent sections for each of these policies later in this chapter describe how each policy implements message protection.

- oracle/wss10_message_protection_client_policy
- oracle/wss10_message_protection_service_policy
- oracle/wss10_username_id_propagation_with_msg_protection_client_policy
- oracle/wss10_username_id_propagation_with_msg_protection_service_policy
- oracle/wss10_username_token_with_message_protection_client_policy
- oracle/wss10_username_token_with_message_protection_service_policy
- oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy
- oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy
- oracle/wss10_x509_token_with_message_protection_client_policy
- oracle/wss10_x509_token_with_message_protection_service_policy
- oracle/wss10_saml_token_with_message_protection_client_policy

- oracle/wss10_saml_token_with_message_protection_service_policy
- oracle/wss10_saml20_token_with_message_protection_client_policy
- oracle/wss10_saml20_token_with_message_protection_service_policy
- oracle/wss10_saml_hok_token_with_message_protection_client_policy
- oracle/wss10_saml_hok_token_with_message_protection_service_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy
- oracle/wss11_message_protection_client_policy
- oracle/wss11_message_protection_service_policy
- oracle/wss11_kerberos_token_with_message_protection_client_policy
- oracle/wss11_kerberos_token_with_message_protection_service_policy
- oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy
- oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy
- oracle/wss11_saml_token_with_message_protection_client_policy
- oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy
- oracle/wss11_saml_token_with_message_protection_service_policy
- oracle/wss11_saml20_token_with_message_protection_client_policy
- oracle/wss11_saml20_token_with_message_protection_service_policy
- oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy
- oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy
- oracle/wss11_sts_issued_saml_with_message_protection_client_policy
- oracle/wss11_username_token_with_message_protection_client_policy
- oracle/wss11_username_token_with_message_protection_service_policy
- oracle/wss11_x509_token_with_message_protection_client_policy
- oracle/wss11_x509_token_with_message_protection_service_policy

Both the WS-Security 1.0 and WS-Security 1.1 standards are supported. Use the assertion template or predefined policy that supports the standard which both the Web service and client share in common. If you are starting anew, use the WS-Security 1.1 standard because it provides more options and requires less PKI deployment.

The assertion templates support partial signing and encryption as well as full signing and encryption of the message body. For those assertion templates or predefined policies that provide SOAP message protection, the default behavior is to protect the entire SOAP message body by signing and encrypting the entire SOAP body. You can configure the assertions and policies to protect selected elements, if you wish.

11.3 Authentication-Only Policies and Configuration Steps

"Authentication Only Policies" on page B-1 summarizes the security policies that enforce authentication only, and indicates whether the token is inserted at the transport layer or header.

This section lists the authentication-only predefined policies, indicates the type of Web service to which they apply, and provides a link to the configuration steps you must perform to use them.

11.3.1 oracle/http_jwt_token_client_policy

The http_jwt_token_client_policy includes a JWT token in the HTTP header. When the policy is used by the client, the JWT token is automatically created by Oracle WSM. The issuer name and subject name are provided either programmatically or declaratively through the policy. You can specify the audience restriction condition for this policy.

This policy can be applied to any HTTP-based client endpoint.

You must edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_jwt_token_client_template" on page C-4 for information about the assertion attributes that you can configure.

By default, the oracle/http_jwt_token_client_policy assertion content is defined as follows:

```
<orasp:http-jwt-security orawsp:Enforced="true" orawsp:Silent="false"
   orawsp:category="security/authentication"
   orawsp:name="Http JWT Security">
   <orasp:auth-header orasp:algorithm-suite="Basic128Sha256Rsa15"
     orasp:is-encrypted="false" orasp:is-signed="true" orasp:mechanism="jwt"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative"
       orawsp:name="HttpJwtTokenConfig">
        <orawsp:PropertySet orawsp:name="standard-security-properties">
          <orawsp:Property orawsp:contentType="optional"
orawsp:name="user.attributes" orawsp:type="string"/>
          <orawsp:Property orawsp:contentType="optional" orawsp:name="issuer.name"
orawsp:type="string">
             <orawsp:Value>www.oracle.com</orawsp:Value>
          </orawsp:Property>
          <orawsp:Property orawsp:contentType="optional"
orawsp:name="user.roles.include" orawsp:type="string">
              <orawsp:Value>false</orawsp:Value>
          </orawsp:Property>
          <orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map"
orawsp:type="string"/>
          <orawsp:Property orawsp:contentType="optional" orawsp:name="csf-key"
orawsp:type="string">
              <orawsp:Value>basic.credentials</orawsp:Value>
          </orawsp:Property>
          <orawsp:Property orawsp:contentType="optional"
orawsp:name="subject.precedence" orawsp:type="string">
              <orawsp:Value>true</orawsp:Value>
          </orawsp:Property>
          <orawsp:Property orawsp:contentType="optional"
 orawsp:name="audience.uri" orawsp:type="string">
              <orawsp:Value/>
          </orawsp:Property>
          <orawsp:Property orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key" orawsp:type="string">
              <orawsp:Value/>
          </orawsp:Property>
          <orawsp:Property orawsp:contentType="optional"
orawsp:name="propagate.identity.context" orawsp:type="string">
              <orawsp:Value/>
          </orawsp:Property>
          <orawsp:Property orawsp:contentType="optional"
orawsp:name="user.tenant.name" orawsp:type="string">
              <orawsp:Value/>
          </orawsp:Property>
          <orawsp:Property orawsp:contentType="optional"
orawsp:name="reference.priority" orawsp:type="string"/>
       </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:http-jwt-security>
```

11.3.1.1 Settings

See Table C–2.

11.3.1.2 Configuration Properties

Table C–3

11.3.2 oracle/http_jwt_token_identity_switch_client_policy

Performs dynamic identity switching by propagating a different identity than the one based on the authenticated subject. This policy includes a JWT token in the HTTP header. When the policy is used by the client, the JWT token is automatically created by Oracle WSM. The issuer name and subject name are provided either programmatically or declaratively through the policy. You can specify the audience restriction condition for this policy.

This policy can be enforced on any HTTP-based, SOAP, or REST client endpoint.

You must edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_jwt_token_client_template" on page C-4 for information about the assertion attributes that you can configure.

By default, the oracle/http_jwt_token_identity_switch_client_policy assertion content is the same as the "oracle/http_jwt_token_client_policy" on page 11-7, except that the `subject.precedence` property is set to `false` as follows:

```
          <orawsp:Property orawsp:contentType="optional"
orawsp:name="subject.precedence" orawsp:type="string">
              <orawsp:Value>true</orawsp:Value>
          </orawsp:Property>
```

11.3.2.1 Settings

See Table C–2.

11.3.2.2 Configuration Properties

Table C–3

11.3.2.3 How to Set Up the Web Service Client

You attach the wss_saml_token_bearer_identity_switch_client_policy to a Web service client of any type.

Override the configuration properties defined in Table C–33, " wss_saml_token_bearer_client_template Configurations". For more information, see "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

subject.precedence is set to false to allow for the use of a client-specified username rather than the authenticated subject. (If subject.precedence is false, the user name to create the SAML assertion is obtained only from the username property of the csf-key.) The wss_saml_token_bearer_identity_switch_client_policy requires that an application to which the policy is attached must have the WSIdentityPermission permission. That is, applications from which Oracle WSM accepts the externally-supplied identity must have the WSIdentityPermission permission. This is to avoid potentially rogue applications from providing an identity to Oracle WSM.

See "Configuring Web Service Clients for Identity Switching" on page 10-78 for information about how to use this policy, including learning "How the Username Is Picked Up by an Identity Switch Policy on the Client Side" on page 10-79 and granting WSIdentityPermission permission, as described in "Set the WSIdentityPermission Permission" on page 10-80.

For additional JWT considerations, see "Using JSON Web Token (JWT) with Oracle WSM" on page 10-69.

11.3.3 oracle/http_jwt_token_service_policy

The http_jwt_token_service_policy authenticates users using the username provided in the JWT token in the HTTP header. By default the policy is configured to expect the JWT token to be signed using the asymmetric signature (algorithm-suite attribute set to Basic128Sha256Rsa15).

You can attach this policy to any HTTP-based endpoint.

You must edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_jwt_token_service_template" on page C-8 for information about the assertion attributes that you can configure.

By default, the oracle/http_jwt_token_service_policy assertion content is defined as follows:

```
<orasp:http-jwt-security orawsp:Enforced="true" orawsp:Silent="false"
   orawsp:category="security/authentication" orawsp:name="Http JWT Security">
   <orasp:auth-header orasp:algorithm-suite="Basic128Sha256Rsa15"
     orasp:is-encrypted="false" orasp:is-signed="true" orasp:mechanism="jwt"/>
   <orawsp:bindings>
       <orawsp:Config orawsp:configType="declarative" orawsp:name="HttpJwtConfig">
          <orawsp:PropertySet orawsp:name="standard-security-properties">
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="trusted.issuers" orawsp:type="string">
                <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map"
orawsp:type="string"/>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key" orawsp:type="string">
                <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="propagate.identity.context" orawsp:type="string">
               <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="reference.priority" orawsp:type="string"/>
```

```
         </orawsp:PropertySet>
      </orawsp:Config>
   </orawsp:bindings>
</orasp:http-jwt-security>
```

11.3.3.1 Settings

See Table C–2.

11.3.3.2 Configuration Properties

See Table C–4.

11.3.4 oracle/http_oam_token_service_policy

This policy extracts the credentials in the HTTP header to authenticate users against the Oracle Platform Security Services identity store and propagates that information using an Oracle Access Manager (OAM) token.

You must edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_oam_token_service_template" on page C-10 for information about the assertion attributes that you can configure.

By default, the oracle/http_oam_token_service_policy assertion content is defined as follows:

```
<orasp:http-oam-security
   xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
   xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
   orawsp:Enforced="true" orawsp:Silent="true"
   orawsp:category="security/authentication" orawsp:name="Http OAM Security">
   <orasp:auth-header orasp:mechanism="oam"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative"
         orawsp:name="HttpOAMConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="reference.priority" orawsp:type="string"/>
         </orawsp:PropertySet>
      </orawsp:Config>
   </orawsp:bindings>
</orasp:http-oam-security>
```

11.3.4.1 Settings

See Table C–5.

11.3.4.2 Configuration Properties

See Table C–6.

11.3.4.3 How to Set Up OAM

You must set up OAM on the server-side. To enforce HTTP OAM security, configure OAM Webgate to intercept the request, authenticate the user, and set the `OAM_REMOTE_USER HTTP` header. Oracle WSM verifies that the `OAM_REMOTE_USER_HTTP` header is present before allowing the request. For more information, see *Administrator's Guide for Oracle Access Management*.

11.3.5 oracle/http_oauth2_token_client_policy

This policy includes the OAuth2 access token in the HTTP header. The access token (AT) is obtained from the Mobile & Social OAuth2 Server. You can attach this policy to any HTTP-based SOAP or REST client.

See "Prerequisites for Using the Oracle WSM OAuth2 Policies" on page 10-76.

You can override the following properties when you attach the policy:

- For OAuth2 token request:
  - scope
  - authz.code (Not used in this release.)
  - redirect.uri (Not used in this release.)
- For local token creation:
  - subject.precedence
  - csf.map
  - csf-key
  - oauth2.client.csf.key
  - federated.client.token
  - user.attributes
  - issuer.name
  - oracle.oauth2.service
  - user.roles.include
  - keystore.sig.csf.key
  - propagate.identity.context
  - user.tenant.name
  - include.certificate
- General:
  - audience.uri
  - reference.priority
  - time.in.millis

You must use WLST or edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_oauth2_token_client_template" on page C-11 for information about the assertion attributes that you can configure.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

You attach this policy and the oracle/oauth2_config_client_policy to the client application.

The required `token.uri` property of the oracle/oauth2_config_client_policy policy specifies the OAuth2 server token endpoint.

You also attach any of the following Oracle WSM JWT service policies to the web service. The Oracle WSM server-side agent validates the access token.

- oracle/http_jwt_token_service_policy

- oracle/multi_token_rest_service_policy (REST)
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy (SOAP)

By default, the oracle/http_oauth2_token_client_policy assertion content is defined as follows:

```
<orasp:http-oauth2-security
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication" orawsp:name="Http OAuth2">
<orasp:auth-header orasp:is-encrypted="false" orasp:is-signed="false"
 orasp:mechanism="oauth2"/>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative" orawsp:name="HttpOAuth2Config">
<orawsp:PropertySet orawsp:name="standard-security-properties">
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="subject.precedence">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="csf.map"/>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="csf-key">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="oauth2.client.csf.key">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
                </orawsp:Property>
                 <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="federated.client.token">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
                 </orawsp:Property>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="scope">
                    <orawsp:Value/>
                </orawsp:Property>
              <orawsp:Property orawsp:type="string"
 orawsp:contentType="optional"
 orawsp:name="authz.code">
                    <orawsp:Value/>
                </orawsp:Property>
                <orawsp:Property orawsp:type="string"
 orawsp:contentType="optional" orawsp:name="redirect.uri">
                    <orawsp:Value/>
                </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="user.attributes">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="issuer.name">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>www.oracle.com</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="oracle.oauth2.service">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="user.roles.include">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="reference.priority">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:name="propagate.identity.context"
orawsp:type="string" orawsp:contentType="optional">
                  <orawsp:Value></orawsp:Value>
                </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="user.tenant.name">
                 <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string"
 orawsp:contentType="optional" orawsp:name="audience.uri">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="include.certificate">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="time.in.millis">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
      </orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:http-oauth2-security>
```

11.3.5.1 Settings

See Table C–7.

11.3.5.2 Configuration Properties

See Table C–8.

11.3.6 oracle/http_oauth2_token_opc_oauth2_client_policy

This policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the OAuth Server in the Oracle Cloud.

The property `oracle.oauth2.service` is set to true by default, which ensures that the client ID is used as the issuer for the user and client JWT tokens for the OAuth2 server. If `scope` is empty (the default), Oracle WSM automatically gets the service URL and uses the address:port portion as the scope.

This policy can be attached to any HTTP-based, SOAP or REST client.

See "Prerequisites for Using the Oracle WSM OAuth2 Policies" on page 10-76.

You can override the following properties when you attach the policy:

- For OAuth2 token request:
  - scope
  - authz.code (Not used in this release.)
  - redirect.uri (Not used in this release.)
- For local token creation:
  - subject.precedence
  - csf.map
  - csf-key
  - oauth2.client.csf.key
  - federated.client.token
  - user.attributes
  - issuer.name
  - oracle.oauth2.service
  - user.roles.include
  - keystore.sig.csf.key
  - propagate.identity.context
  - user.tenant.name
  - include.certificate
- General:
  - audience.uri
  - reference.priority
  - time.in.millis

You must use WLST or edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_oauth2_token_client_template" on page C-11 for information about the assertion attributes that you can configure.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

You attach this policy and the oracle/oauth2_config_client_policy to the client application. The required `token.uri` property of the oracle/oauth2_config_client_policy policy specifies the OAuth2 server.

You also attach any of the following Oracle WSM JWT service policies to the web service. The Oracle WSM server-side agent validates the access token.

- oracle/http_jwt_token_service_policy

- oracle/multi_token_rest_service_policy (REST)
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy (SOAP)

By default, the oracle/http_oauth2_token_opc_oauth2_client_policy assertion content is defined as follows:

```
<orasp:http-oauth2-security
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication" orawsp:name="Http OAuth2">
<orasp:auth-header orasp:is-encrypted="false" orasp:is-signed="false"
 orasp:mechanism="oauth2"/>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative" orawsp:name="HttpOAuth2Config">
<orawsp:PropertySet orawsp:name="standard-security-properties">
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="subject.precedence">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="csf.map"/>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="csf-key">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="oauth2.client.csf.key">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
                </orawsp:Property>
                 <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="federated.client.token">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
                 </orawsp:Property>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="scope">
                    <orawsp:Value/>
                </orawsp:Property>
                 <orawsp:Property orawsp:type="string"
 orawsp:contentType="optional" orawsp:name="authz.code">
                    <orawsp:Value/>
                </orawsp:Property>
                <orawsp:Property orawsp:type="string"
 orawsp:contentType="optional" orawsp:name="redirect.uri">
                    <orawsp:Value/>
                </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="user.attributes">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="issuer.name">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="oracle.oauth2.service">
```

```
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="user.roles.include">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="reference.priority">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:name="propagate.identity.context"
orawsp:type="string" orawsp:contentType="optional">
                  <orawsp:Value></orawsp:Value>
                </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="user.tenant.name">
                 <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string"
 orawsp:contentType="optional" orawsp:name="audience.uri">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="include.certificate">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="time.in.millis">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
      </orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:http-oauth2-security>
```

11.3.6.1 Settings

See Table C–7.

11.3.6.2 Configuration Properties

See Table C–8.

11.3.7 oracle/oauth2_config_client_policy

This policy provides OAuth2 information on the client side. This information is used to invoke the Mobile and Social OAuth2 server for token exchange.

This policy is enforced only when an OAuth2 token client policy is also attached. Otherwise, it is ignored.   This policy is typically attached globally, and the OAuth2 token client policy locally.

See "Prerequisites for Using the Oracle WSM OAuth2 Policies" on page 10-76.

You must use WLST or edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/oauth2_config_client_template" on page C-18 for information about the assertion attributes that you can configure.

You must set or override the `token.uri` property.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

By default, the oracle/oauth2_config_client_policy assertion content is defined as follows:

```
<orasp:oauth2-config
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orasp:token-uri="http://host:port/tokens" orawsp:Enforced="true"
 orawsp:Silent="true" orawsp:category="security/oauth2-config"
 orawsp:name="OAuth2 Configuration">
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative" orawsp:name="OAuth2Config">
<orawsp:PropertySet orawsp:name="standard-security-properties">
               <orawsp:Property orawsp:name="role" orawsp:type="string"
orawsp:contentType="constant">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>ultimateReceiver</orawsp:DefaultValue>
               </orawsp:Property>
<orawsp:Property orawsp:name="token.uri" orawsp:type="string"
orawsp:contentType="optional">
                  <orawsp:Value/>
<orawsp:DefaultValue>http://host:port/tokens</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="required"
orawsp:name="oauth2.client.csf.key">
                  <orawsp:Value/>
<orawsp:DefaultValue>basic.client.credentials</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="reference.priority"/>
             </orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:oauth2-config>
```

11.3.7.1 Settings

See Table C–9.

11.3.7.2 Configuration Properties

See Table C–10.

11.3.8 oracle/http_saml20_token_bearer_client_policy

The http_saml20_token_bearer_client_policy policy includes SAML 2.0 tokens in the HTTP header. The SAML token with confirmation method *Bearer* is created automatically. This policy can be enforced on any HTTP-based endpoint. By default, the authenticated user from the Subject (user principal) is used to generate the SAML assertion for identity propagation.

You must edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_saml20_token_bearer_client_template" on page C-20 for information about the assertion attributes that you can configure.

By default, the oracle/http_saml_bearer_token_service_policy assertion content is defined as follows:

```
<orasp:http-saml20-bearer-security
   xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
   xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
   orawsp:Enforced="true" orawsp:Silent="true"
   orawsp:category="security/authentication" orawsp:name="Http SAML 2.0 Bearer
   Security">
   <orasp:auth-header orasp:mechanism="saml20-bearer"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative"
         orawsp:name="HttpSaml20BearerConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="user.attributes" orawsp:type="string"/>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.issuer.name" orawsp:type="string">
               <orawsp:Value>www.oracle.com</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="user.roles.include" orawsp:type="string">
               <orawsp:Value>false</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="csf-key" orawsp:type="string">
               <orawsp:Value>basic.credentials</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="subject.precedence" orawsp:type="string">
               <orawsp:Value>true</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.audience.uri" orawsp:type="string">
               <orawsp:Value/>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="keystore.sig.csf.key" orawsp:type="string"/>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.enveloped.signature.required"
               orawsp:type="boolean">
               <orawsp:Value>true</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="reference.priority" orawsp:type="string"/>
            <orawsp:Property orawsp:name="propagate.identity.context"
               orawsp:type="string" orawsp:contentType="optional">
               <orawsp:Value></orawsp:Value>
         </orawsp:PropertySet>
      </orawsp:Config>
   </orawsp:bindings>
</orasp:http-saml20-bearer-security>
```

11.3.8.1 Settings

See Table C–11.

11.3.8.2 Configuration Properties

See Table C–12.

11.3.9 oracle/http_saml20_bearer_token_service_policy

This policy authenticates users using credentials provided in SAML 2.0 tokens with confirmation method 'Bearer' in the HTTP header.

You must edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_saml20_token_bearer_service_template" on page C-23 for information about the assertion attributes that you can configure.

By default, the oracle/http_saml_bearer_token_over_ssl_service_policy assertion content is defined as follows:

```
<orasp:http-saml20-bearer-security
   xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
   xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
   orawsp:Enforced="true" orawsp:Silent="true" '
   orawsp:category="security/authentication" orawsp:name="Http SAML 2.0 Bearer
   Security">
   <orasp:auth-header orasp:mechanism="saml20-bearer"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative"
         orawsp:name="HttpSaml20BearerConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.trusted.issuers" orawsp:type="string">
               <orawsp:Value/>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.enveloped.signature.required"
               orawsp:type="boolean">
               <orawsp:Value>true</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="reference.priority" orawsp:type="string"/>
            <orawsp:Property orawsp:name="propagate.identity.context"
               orawsp:type="string" orawsp:contentType="optional">
               <orawsp:Value></orawsp:Value>
         </orawsp:PropertySet>
      </orawsp:Config>
   </orawsp:bindings>
</orasp:http-saml20-bearer-security>
```

11.3.9.1 Settings

See Table C–11.

11.3.9.2 Configuration Properties

See Table C–13.

11.3.10 oracle/wss_http_token_client_policy

The oracle/wss_http_token_client_policy policy includes credentials in the HTTP header for outbound client requests.   It is the analogous client policy to the oracle/wss_http_token_service_policy service endpoint policy.

This policy contains the following assertion template: oracle/wss_http_token_client_template. See "oracle/wss_http_token_client_template" on page 27 for more information about the assertion.

11.3.10.1 Settings

See Table C–17.

11.3.10.2 Configuration Properties

See Table C–18.

11.3.10.3 How to Set Up the Web Service Client

You can specify a value for `csf-key` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Adding Keys and User Credentials to the Credential Store" on page 10-19 for information on how to add the key to the credential store.

11.3.10.4 How to Set Up the Web Service Client at Design Time

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

The client must pass the credentials in the HTTP header.

11.3.10.5 How to Attach and Configure the Policy for Servlet Applications

For servlet applications, you must attach and modify the policy file manually; you cannot attach, view, or edit the policy using Fusion Middleware Control.

To attach the policy to a servlet, see "Attaching Policies to Servlet Applications" on page 8-18.

For information about the assertion attributes that you can configure, see "orasp:http-security" on page D-19.

By default, the policy is defined as follows:

```
<orasp:http-security orawsp:Enforced="true" orawsp:Silent="true"
   orawsp:category="security/authentication" orawsp:name="Http Security">
   <orasp:auth-header orasp:mechanism="basic"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative" orawsp:name="HttpConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:name="csf-key" orawsp:type="string">
               <orawsp:Value>basic.credentials</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
               orawsp:type="string">
               <orawsp:Value>ultimateReceiver</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="reference.priority" orawsp:type="string"/>
         </orawsp:PropertySet>
      </orawsp:Config>
   </orawsp:bindings>
</orasp:http-security>
```

11.3.11 oracle/wss_http_token_service_policy

The wss_http_token_service_policy uses the credentials in the HTTP header to authenticate users.

This policy contains the following assertion template: oracle/wss_http_token_service_template. See "oracle/wss_http_token_service_template" on page 29 for more information about the assertion.

11.3.11.1 Settings

See Table C–17.

Note: For servlet applications, see "How to Attach and Configure the Policy for Servlet Applications" on page 11-21.

11.3.11.2 Configuration Properties

See Table C–19.

Note: For servlet applications, see "How to Attach and Configure the Policy for Servlet Applications" on page 11-21.

11.3.11.3 How to Set Up WebLogic Server

The Web service must authenticate the supplied username and password credentials against the configured authentication source.

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

For mutual SSL authentication, you must configure WebLogic Server. See "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40.

11.3.11.4 How to Attach and Configure the Policy for Servlet Applications

For servlet applications, you must attach and modify the policy file manually; you cannot attach, view, or edit the policy using Fusion Middleware Control.

To attach the policy to a servlet, see "Attaching Policies to Servlet Applications" on page 8-18.

For information about the assertion attributes that you can configure, see "orasp:http-security" on page D-19.

By default, the policy is defined as follows:

```
<orasp:http-security orawsp:Enforced="true" orawsp:Silent="true"
   orawsp:category="security/authentication" orawsp:name="Http Security">
   <orasp:auth-header orasp:mechanism="basic"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative" orawsp:name="HttpConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:contentType="constant" orawsp:name="realm"
               orawsp:type="string">
               <orawsp:Value>Oracle WSM</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
```

```
               orawsp:type="string">
               <orawsp:Value>ultimateReceiver</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="reference.priority" orawsp:type="string"/>
         </orawsp:PropertySet>
       </orawsp:Config>
   </orawsp:bindings>
<orasp:http-security>
```

11.3.12 oracle/wss_username_token_client_policy

Note: This policy is not secure; it transmits the password in clear text. You should use this policy in low security situations only, or when you know that the transport is protected using some other mechanism. Alternatively, consider using the SSL version of this policy, oracle/wss_username_token_over_ssl_client_policy.

This policy includes credentials in the WS-Security UsernameToken header for all outbound SOAP request messages. A plain text mechanism is supported, in addition to a password not being required. It is the analogous client policy to the oracle/wss_username_token_service_policy service endpoint policy.

This policy contains the following assertion template: oracle/wss_username_token_client_template. See "oracle/wss_username_token_client_template" on page C-30 for more information about the assertion.

11.3.12.1 Settings

See Table C–20.

11.3.12.2 Configuration Properties

See Table C–21.

11.3.12.3 How to Set Up the Web Service Client

You can specify a value for `csf-key` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Adding Keys and User Credentials to the Credential Store" on page 10-19 for information on how to add the key to the credential store.

If you specify a password type of None on the **Settings** page, you do not need to include a password in the key.

11.3.12.4 How to Set Up the Web Service Client At Design Time

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

The client must include a WS-Security UsernameToken element (<wsse:UsernameToken/>) in the SOAP request message. The client provides a username and password for authentication.

11.3.13 oracle/wss_username_token_service_policy

> **Note:** This policy is not secure; it transmits the password in clear text. You should use this policy in low security situations only, or when you know that the transport is protected using some other mechanism. Alternatively, consider using the SSL version of this policy, oracle/wss_username_token_over_ssl_service_policy.

This policy uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users. The plain text mechanism is supported.

This policy contains the following assertion template: oracle/wss_username_token_service_template. See "oracle/wss_username_token_service_template" on page C-33 for more information about the assertion.

11.3.13.1 Settings

See Table C–20.

11.3.13.2 Configuration Properties

See Table C–22.

11.3.13.3 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.3.14 oracle/wss10_saml_token_client_policy

> **Note:** This policy is not secure and is provided for demonstration purposes only. Although the SAML issuer name is present, the SAML token is not endorsed. Therefore, it is possible to spoof the message.

This policy includes SAML tokens in outbound SOAP request messages.

This policy contains the following assertion template: oracle/wss10_saml_token_client_template. See "oracle/wss10_saml_token_client_template" on page 34 for more information about the assertion.

11.3.14.1 Settings

See Table C–23.

11.3.14.2 Configuration Properties

See Table C–24.

11.3.14.3 How to Set Up the Web Service Client

See "Configuring SAML" on page 10-64.

You can specify a value for `saml.issuer.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `saml.issuer.name` property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.3.14.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Include a WS-Security Header Element (<saml:Assertion>) that inserts a SAML token in the outbound SOAP message. The confirmation type is always *sender-vouches*.

11.3.15 oracle/wss10_saml_token_service_policy

> **Note:** This policy is not secure and is provided for demonstration purposes only. Although the SAML issuer name is present, the SAML token is not endorsed. Therefore, it is possible to spoof the message.

This policy authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.

This policy contains the following assertion template: oracle/wss10_saml_token_service_template. See "oracle/wss10_saml_token_service_template" on page C-38 for more information about the assertion.

11.3.15.1 Settings

See Table C–23.

11.3.15.2 Configuration Properties

See Table C–25.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.3.15.3 Configure the Login Module

Configure the `saml.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

11.3.15.4 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the provider.

11.3.16 oracle/wss10_saml20_token_client_policy

Note: This policy is not secure and is provided for demonstration purposes only. Although the SAML issuer name is present, the SAML token is not endorsed. Therefore, it is possible to spoof the message.

This policy includes SAML tokens in outbound SOAP request messages.

This policy contains the following assertion template: oracle/wss10_saml20_token_client_template. See "oracle/wss10_saml20_token_client_template" on page 39 for more information about the assertion.

11.3.16.1 Settings

See Table C–26.

11.3.16.2 Configuration Properties

See Table C–27.

11.3.16.3 How to Set Up the Web Service Client

See "Configuring SAML" on page 10-64.

You can specify a value for `saml.issuer.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `saml.issuer.name` property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.3.16.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Include a WS-Security Header Element (<saml:Assertion>) that inserts a SAML token in the outbound SOAP message. The confirmation type is always *sender-vouches*.

11.3.17 oracle/wss10_saml20_token_service_policy

Note: This policy is not secure and is provided for demonstration purposes only. Although the SAML issuer name is present, the SAML token is not endorsed. Therefore, it is possible to spoof the message.

This policy authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.

This policy contains the following assertion template: oracle/wss10_saml20_token_service_template. See "oracle/wss10_saml20_token_service_template" on page C-43 for more information about the assertion.

11.3.17.1 Settings

See Table C–26.

11.3.17.2 Configuration Properties

See Table C–28.

You can also specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.3.17.3 Configure the Login Module

Configure the `saml2.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

11.3.17.4 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the provider.

11.3.18 oracle/wss11_kerberos_token_client_policy

This policy includes a Kerberos token in the WS-Security header in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

Service principal names (SPN) are a key component in Kerberos authentication. SPNs are unique identifiers for services running on servers. Every service that uses Kerberos authentication needs to have an SPN set for it so that clients can identify the service on the network. If an SPN is not set for a service, clients have no way of locating that service and Kerberos authentication is not possible.

This policy contains the following assertion template: oracle/wss11_kerberos_token_client_template. See "oracle/wss11_kerberos_token_with_message_protection_client_ template" on page C-134 for more information about the assertion.

11.3.18.1 Settings

See Table C–81.

11.3.18.2 Configuration Properties

See Table C–82.

11.3.18.3 How to Set Up the Web Service Client

See "Using Kerberos Tokens" on page 10-85.

The Web service client that is enforcing Kerberos client side policies needs to know the service principal name of the service it is trying to access. You can specify a value for `service.principal.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The default value (place holder) is `HOST/localhost@oracle.com`.

11.3.18.4 How to Set Up the Web Service Client at Design Time

See "Using Kerberos Tokens" on page 10-85.

You must set the service principal name. The service principal name specifies the name of the service principal for which the client requests a ticket from the KDC.

If the Kerberos authentication is successful, then send the obtained Kerberos ticket and authenticator to the Web service enclosed in a BinarySecurityToken element in the SOAP Security header.

11.3.19 oracle/wss11_kerberos_token_service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

Service principal names (SPN) are a key component in Kerberos authentication. SPNs are unique identifiers for services running on servers. Every service that uses Kerberos authentication needs to have an SPN set for it so that clients can identify the service on the network. If an SPN is not set for a service, clients have no way of locating that service and Kerberos authentication is not possible.

This policy contains the following assertion template: oracle/wss11_kerberos_token_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-137 for more information about the assertion.

11.3.19.1 Settings

See Table C–81.

11.3.19.2 Configuration Properties

None required.

11.3.19.3 Configure the Login Module

Configure the `krb5.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

11.3.19.4 How to Configure WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.3.20 oracle/wss_saml_token_bearer_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method *Bearer* is created automatically.

This policy contains the following assertion template: oracle/wss_saml_token_bearer_client_template. See "oracle/wss_saml_token_bearer_client_template" on page C-46 for more information about the assertion.

11.3.20.1 Settings

See Table C–32

11.3.20.2 Configuration Properties

See Table C–33

11.3.20.3 How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

11.3.20.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

11.4 Message Protection-Only Policies and Configuration Steps

See "Protecting Messages" on page 11-2 for a description of how the predefined policies implement message protection.

Table B–3 summarizes the policies that enforce only message protection, and indicates whether the policy is enforced at the transport layer or SOAP header.

Message protection-only policies do not authenticate or authorize the requester.

There may be either one or two Security policies attached to a policy subject. A Security policy can contain an assertion that belongs to the authentication or message protection (as in this case) subtype categories, or a single assertion that belongs to both subtype categories. You can then use an assertion that belongs to the authorization subtype to authorize the requester.

11.4.1 oracle/wss10_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_message_protection_client_template. See "oracle/wss10_message_protection_client_policy" on page B-9 for more information about the assertion.

11.4.1.1 Settings

See Table C–35.

11.4.1.2 Configuration Properties

See Table C–36.

11.4.1.3 How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `keystore.recipient.alias` specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

11.4.1.4 How to Set Up the Web Service Client at Design Time

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–3 WS-Security 1.0 Message Integrity of SOAP Message

```
<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
 <dsig:SignedInfo>
  <dsig:CanonicalizationMethod
   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <dsig:Reference URI="#Timestamp-...">
     <dsig:Transforms>
       <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
     </dsig:Transforms>
     <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <dsig:DigestValue>...</dsig:DigestValue>
  </dsig:Reference>
  <dsig:Reference URI="#Body-...">
     <dsig:Transforms>
         <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
     </dsig:Transforms>
```

```
      <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <dsig:DigestValue>...</dsig:DigestValue>
   </dsig:Reference>
   <dsig:Reference URI="#KeyInfo-...">
    <dsig:Transforms>
      <dsig:Transform
Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-se
curity-1.0#STR-Transform">
        <TransformationParameters
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1
.0.xsd">
        <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
xmlns="http://www.w3.org/2000/09/xmldsig#"/>
        </TransformationParameters>
      </dsig:Transform>
    </dsig:Transforms>
    <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <dsig:DigestValue>...</dsig:DigestValue>
   </dsig:Reference>
 </dsig:SignedInfo>
 <dsig:SignatureValue>....</dsig:SignatureValue>
 <dsig:KeyInfo Id="KeyInfo-...">
     <wsse:SecurityTokenReference
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1
.0.xsd">
      <wsse:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-prof
ile-1.0#X509SubjectKeyIdentifier"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message
-security-1.0#Base64Binary">
...</wsse:KeyIdentifier>
     </wsse:SecurityTokenReference>
 </dsig:KeyInfo>
</dsig:Signature>
```

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

Example 11–4 WS-Security 1.0 Message Confidentiality of SOAP Message

```
<env:Body
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd" wsu:Id="Body-JA9fsCRnqbFJ0ocBAMKb7g22">
 <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Content" Id="...">
  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
  <xenc:CipherData>
      <xenc:CipherValue>...</xenc:CipherValue>
  </xenc:CipherData>
 </xenc:EncryptedData>
</env:Body>
```

11.4.2 oracle/wss10_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

The messages are protected using WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. This policy does not authenticate or authorize the requester.

This policy contains the following assertion template: oracle/wss10_message_protection_service_template. See "oracle/wss10_message_protection_service_template" on page C-53 for more information about the assertion.

11.4.2.1 Settings

See Table C–35.

11.4.2.2 Configuration Properties

See Table C–37.   You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.4.2.3 How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.4.3 oracle/wss11_message_protection_client_policy

This policy provides message integrity and confidentiality for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following assertion template: oracle/wss11_message_protection_client_template. See "oracle/wss11_message_protection_client_template" on page C-55 for more information about the assertion.

11.4.3.1 Settings

See Table C–38.

11.4.3.2 Configuration Properties

See Table C–39.

11.4.3.3 How to Configure the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for keystore.recipient.alias on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for keystore.enc.csf.key on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

11.4.3.4 How to Configure the Web Service Client at Design Time

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

This policy uses symmetric key technology, which is an encryption method that uses the same shared key to encrypt and decrypt data. The symmetric key is used to sign the message.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–5 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.1 standards. In this example, the body element is encrypted.

Example 11–5 WS-Security 1.1 Message Confidentiality of SOAP Message

```
<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="EK-...">
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
<dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" />
</xenc:EncryptionMethod>
<dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
<wsse:SecurityTokenReference
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1
.0.xsd">
<wsse:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#Thum
bprintSHA1"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message
-security-1.0#Base64Binary">...</wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
</dsig:KeyInfo>
<xenc:CipherData>
<xenc:CipherValue>...</xenc:CipherValue>
</xenc:CipherData>
<xenc:ReferenceList>
<xenc:DataReference URI="#_..." />
</xenc:ReferenceList>
</xenc:EncryptedKey>
<env:Body
```

```
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd" wsu:Id="Body-...">
  <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Content" Id="...">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"
/>
    <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <wsse:SecurityTokenReference
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-sec
ext-1.0.xsd"
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1
.0.xsd">
         <wsse:Reference URI="#EK-..."
ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#Encr
yptedKey" />
      </wsse:SecurityTokenReference>
    </dsig:KeyInfo>
    <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</env:Body>
```

11.4.4 oracle/wss11_message_protection_service_policy

This policy enforces message integrity and confidentiality for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following assertion template: oracle/wss11_message_protection_service_template. See "oracle/wss11_message_protection_service_template" on page C-58 for more information about the assertion.

11.4.4.1 Settings

See Table C–38.

11.4.4.2 Configuration Properties

See Table C–40. You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.4.4.3 How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5 Message Protection and Authentication Policies and Configuration Steps

Table B–4 summarizes the policies that enforce both message protection and authentication, and indicates whether the policy is enforced at the transport layer or SOAP header. These polices are described in the sections that follow.

See "Protecting Messages" on page 11-2 for a description of how the predefined policies implement message protection.

11.5.1 Configuring a Policy With an OR Group

The following policies contain assertions as an OR group—meaning that either type of assertion can be enforced by a client:

- oracle/multi_token_rest_service_policy
- oracle/multi_token_over_ssl_rest_service_policy
- oracle/wss_saml_or_username_token_over_ssl_service_policy
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy

In addition, you can add an OR group to the policy of your choice, as described in "Adding an OR Group to a Policy" on page 7-13.

The oracle/multi_token_rest_service_policy policy contains the following assertion templates:

- oracle/wss_http_token_service_template. For more information about configuring the policy, see "oracle/wss_http_token_service_policy" on page 11-21.
- oracle/http_saml20_token_bearer_service_template. For more information about configuring the policy, see "oracle/http_saml20_bearer_token_service_policy" on page 11-19.
- oracle/http_oam_token_service_template. For more information about configuring the policy, see "oracle/http_oam_token_service_policy" on page 11-10.
- oracle/http_spnego_token_service_template. For more information about configuring a policy generated using this assertion template, see "Configuring Kerberos With SPNEGO Negotiation" on page 10-89.
- oracle/http_jwt_token_service_template. For more information about configuring the policy, see "oracle/http_jwt_token_service_policy" on page 11-9.

The oracle/multi_token_over_ssl_rest_service_policy policy contains the following assertion templates:

- oracle/http_basic_auth_over_ssl_service_template. For more information about configuring the policy, see "oracle/http_basic_auth_over_ssl_service_policy" on page 11-36.
- oracle/http_saml20_token_bearer_over_ssl_service_template. For more information about configuring the policy, see "oracle/http_saml20_bearer_token_over_ssl_service_policy" on page 11-54.
- oracle/http_oam_token_service_template. For more information about configuring the policy, see "oracle/http_oam_token_service_policy" on page 11-10.
- oracle/http_spnego_token_service_template. For more information about configuring a policy generated using this assertion template, see "Configuring Kerberos With SPNEGO Negotiation" on page 10-89.

- oracle/http_jwt_token_over_ssl_service_template. For more information about configuring the policy, see "oracle/http_jwt_token_over_ssl_service_policy" on page 11-39.

The oracle/wss_saml_or_username_token_over_ssl_service_policy policy contains the following assertion templates:

- oracle/wss_saml_token_over_ssl_service_template. For information about configuring the policy, see "oracle/wss_saml_token_over_ssl_service_policy" on page 11-61.
- oracle/wss_username_token_over_ssl_service_template. For information about configuring the policy, see "oracle/wss_username_token_over_ssl_service_policy" on page 11-64.

The oracle/wss11_saml_or_username_token_with_message_protection_service_policy contains the following assertion templates:

- oracle/wss11_saml_token_with_message_protection_service_template. For information about configuring the policy, see "oracle/wss11_saml_token_with_message_protection_service_policy" on page 11-91.
- oracle/wss11_username_token_with_message_protection_service_template. For information about configuring the policy, see "oracle/wss11_username_token_with_message_protection_service_policy" on page 11-96.
- oracle/wss_saml_token_bearer_over_ssl_service_template. For information about configuring the policy, see "oracle/wss_saml_token_bearer_over_ssl_service_policy" on page 11-58.
- oracle/wss_username_token_over_ssl_service_template. For information about configuring the policy, see "oracle/wss_username_token_over_ssl_service_policy" on page 11-64.
- oracle/wss_http_token_over_ssl_service_template. For information about configuring the policy, see "oracle/wss_http_token_over_ssl_service_policy" on page 11-56.
- oracle/http_jwt_token_over_ssl_service_template. For more information about configuring the policy, see "oracle/http_jwt_token_over_ssl_service_policy" on page 11-39.

11.5.2 oracle/http_basic_auth_over_ssl_client_policy

The http_basic_auth_over_ssl_client_policy policy includes credentials in the HTTP header for outbound client requests.

This policy also verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be applied to any HTTP-based endpoint.

Note: Currently only HTTP basic authentication is supported.

11.5.2.1 Settings

See Table C–46.

11.5.2.2 Configuration Properties

See Table C–47.

11.5.2.3 How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40.

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.5.2.4 How to Attach and Configure the Policy for Servlet Applications

For servlet applications, you must attach and modify the policy file manually; you cannot attach, view, or edit the policy using Fusion Middleware Control.

To attach the policy to a servlet, see "Attaching Policies to Servlet Applications" on page 8-18.

For information about the assertion attributes that you can configure, see "orasp:http-security" on page D-19.

By default, the policy is defined as follows:

```
<orasp:http-security orawsp:Enforced="true" orawsp:Silent="true"
   orawsp:category="security/authentication, security/msg-protection"
   orawsp:name="Http over SSL Security">
   <orasp:auth-header orasp:mechanism="basic"/>
   <orasp:require-tls orasp:include-timestamp="false"
      orasp:mutual-auth="false"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative" orawsp:name="HttpConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:name="csf-key" orawsp:type="string">
               <orawsp:Value>basic.credentials</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="reference.priority" orawsp:type="string"/>
         </orawsp:PropertySet>
      </orawsp:Config>
   </orawsp:bindings>
</orasp:http-security>
```

11.5.3 oracle/http_basic_auth_over_ssl_service_policy

The http_basic_auth_over_ssl_service_policy policy extracts the credentials in the HTTP header and authenticates users against the Oracle Platform Security Services identity store. This policy verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be enforced on any HTTP-based endpoint.

Note: Currently only HTTP basic authentication is supported.

11.5.3.1 Settings

See Table C–46.

11.5.3.2 Configuration Properties

See Table C–48.

11.5.3.3 How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40.

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.5.3.4 How to Attach and Configure the Policy for Servlet Applications

For servlet applications, you must attach and modify the policy file manually; you cannot attach, view, or edit the policy using Fusion Middleware Control.

To attach the policy to a servlet, see "Attaching Policies to Servlet Applications" on page 8-18.

For information about the assertion attributes that you can configure, see "orasp:http-security" on page D-19.

By default, the policy is defined as follows:

```
<orasp:http-security orawsp:Enforced="true" orawsp:Silent="true"
   orawsp:category="security/authentication, security/msg-protection"
   orawsp:name="Http over SSL Security">
   <orasp:auth-header orasp:mechanism="basic"/>
   <orasp:require-tls orasp:include-timestamp="false"
      orasp:mutual-auth="false"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative" orawsp:name="HttpConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:name="csf-key" orawsp:type="string">
               <orawsp:Value>basic.credentials</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="reference.priority" orawsp:type="string"/>
         </orawsp:PropertySet>
      </orawsp:Config>
   </orawsp:bindings>
</orasp:http-security>
```

11.5.4 oracle/http_jwt_token_over_ssl_client_policy

The http_jwt_token_over_ssl_client_policy includes a JWT token in the HTTP header. When the policy is used by the client, the JWT token is automatically created by Oracle WSM. The issuer name and subject name are provided either programmatically or declaratively through the policy. You can specify the audience restriction condition for this policy.

This policy verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused.

This policy can be applied to any HTTP-based client endpoint.

You must edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_jwt_token_over_ssl_client_template" on page C-62 for information about the assertion attributes that you can configure.

By default, the oracle/http_jwt_token_over_ssl_client_policy assertion content is defined as follows:

```
<orasp:http-jwt-security orawsp:Enforced="true" orawsp:Silent="false"
    orawsp:category="security/authentication" orawsp:name="Http JWT Security">
    <orasp:auth-header orasp:algorithm-suite="Basic128Sha256Rsa15"
      orasp:is-encrypted="false" orasp:is-signed="true" orasp:mechanism="jwt"/>
    <orasp:require-tls orasp:include-timestamp="false" orasp:mutual-auth="false"/>
    <orawsp:bindings>
        <orawsp:Config orawsp:configType="declarative"
orawsp:name="HttpJwtTokenConfig">
          <orawsp:PropertySet orawsp:name="standard-security-properties">
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="user.attributes" orawsp:type="string"/>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="issuer.name" orawsp:type="string">
                <orawsp:Value>www.oracle.com</orawsp:Value>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="user.roles.include" orawsp:type="string">
                <orawsp:Value>false</orawsp:Value>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map"
orawsp:type="string"/>
             <orawsp:Property orawsp:contentType="optional" orawsp:name="csf-key"
orawsp:type="string">
                <orawsp:Value>basic.credentials</orawsp:Value>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="subject.precedence" orawsp:type="string">
                <orawsp:Value>true</orawsp:Value>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
 orawsp:name="audience.uri" orawsp:type="string">
                <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key" orawsp:type="string">
                <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="propagate.identity.context" orawsp:type="string">
                 <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="user.tenant.name" orawsp:type="string">
                <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="reference.priority" orawsp:type="string"/>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:http-jwt-security>
```

11.5.4.1 Settings

See Table C–42.

11.5.4.2 Configuration Properties

Table C–43

11.5.4.3 How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40.

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.5.5 oracle/http_jwt_token_over_ssl_service_policy

The http_jwt_token_service_policy authenticates users using the username provided in the JWT token in the HTTP header. By default the policy is configured to expect the JWT token to be signed using the asymmetric signature (`algorithm-suite` attribute set to `Basic128Sha256Rsa15`).

This policy also verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be applied to any HTTP-based endpoint.

You must edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_jwt_token_over_ssl_service_template" on page C-67 for information about the assertion attributes that you can configure.

By default, the oracle/http_jwt_token_over_ssl_service_policy assertion content is defined as follows:

```
<orasp:http-jwt-security orawsp:Enforced="true" orawsp:Silent="false"
   orawsp:category="security/authentication" orawsp:name="Http JWT Security">
   <orasp:auth-header orasp:algorithm-suite="Basic128Sha256Rsa15"
     orasp:is-encrypted="false" orasp:is-signed="true" orasp:mechanism="jwt"/>
   <orasp:require-tls orasp:include-timestamp="false" orasp:mutual-auth="false"/>
   <orawsp:bindings>
       <orawsp:Config orawsp:configType="declarative" orawsp:name="HttpJwtConfig">
          <orawsp:PropertySet orawsp:name="standard-security-properties">
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="trusted.issuers" orawsp:type="string">
                <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map"
orawsp:type="string"/>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key" orawsp:type="string">
                <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="propagate.identity.context" orawsp:type="string">
               <orawsp:Value/>
             </orawsp:Property>
             <orawsp:Property orawsp:contentType="optional"
orawsp:name="reference.priority" orawsp:type="string"/>
         </orawsp:PropertySet>
     </orawsp:Config>
   </orawsp:bindings>
</orasp:http-jwt-security>
```

11.5.5.1 Settings

See Table C–42.

11.5.5.2 Configuration Properties

See Table C–44.

11.5.5.3 How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40.

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.5.6 oracle/http_oauth2_token_over_ssl_client_policy

This policy is the same as http_oauth2_token_client_policy, except that the AT is propagated over 1-way SSL to the resource. This policy includes the OAauth2 access token in the HTTP header. The AT is obtained from the Mobile and Social OAuth2 Server.

The policy verifies that the outbound transport protocol is HTTPS. If a non-HTTPS transport protocol is used, the request is refused. You can attach this policy to any HTTP-based client.

See "Prerequisites for Using the Oracle WSM OAuth2 Policies" on page 10-76.

You can override the following properties when you attach the policy:

- For OAuth2 token request:
  - scope
  - authz.code (Not used in this release.)
  - redirect.uri (Not used in this release.)
- For local token creation:
  - subject.precedence
  - csf.map
  - csf-key
  - oauth2.client.csf.key
  - federated.client.token
  - user.attributes
  - issuer.name
  - oracle.oauth2.service
  - user.roles.include
  - keystore.sig.csf.key
  - propagate.identity.context
  - user.tenant.name
  - include.certificate
- General:
  - audience.uri

- reference.priority
- time.in.millis

You must use WLST or edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_oauth2_token_over_ssl_client_template" on page C-69 for information about the assertion attributes that you can configure.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

You attach this policy and the oracle/oauth2_config_client_policy to the client application.   The required `token.uri` property of the oracle/oauth2_config_client_policy policy specifies the OAuth2 server.

You also attach any of the following Oracle WSM JWT service policies to the web service. The Oracle WSM server-side agent validates the AT.

- oracle/http_jwt_token_over_ssl_service_policy
- oracle/multi_token_over_ssl_rest_service_policy (REST)
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy (SOAP)

By default, the oracle/http_oauth2_token_over_ssl_client_policy assertion content is defined as follows:

```
<orasp:http-oauth2-security
xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="Http OAuth2 Over SSL ">
<orasp:auth-header orasp:is-encrypted="false" orasp:is-signed="false"
orasp:mechanism="oauth2"/>
<orasp:require-tls orasp:algorithm-suite="Basic128"
orasp:include-timestamp="false" orasp:mutual-auth="false"/>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
orawsp:name="HttpOAuth2OverSSLConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="subject.precedence">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="csf.map"/>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="csf-key">
                  <orawsp:Value/>
               </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="oauth2.client.csf.key">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
                </orawsp:Property>
                 <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="federated.client.token">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
                 </orawsp:Property>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="scope">
                    <orawsp:Value/>
                </orawsp:Property>
orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="authz.code">
                <orawsp:Value/>
                </orawsp:Property>
orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="redirect.uri">
                    <orawsp:Value/>
                </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="user.attributes">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="issuer.name">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>www.oracle.com</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="oracle.oauth2.service">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="user.roles.include">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="reference.priority">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:name="propagate.identity.context"
orawsp:type="string" orawsp:contentType="optional">
                  <orawsp:Value></orawsp:Value>
                </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="user.tenant.name">
                   <orawsp:Value/>
               </orawsp:Property>
<orawsp:Property orawsp:type="string" orawsp:contentType="optional"
 orawsp:name="audience.uri">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="include.certificate">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="time.in.millis">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
      </orawsp:PropertySet>
         </orawsp:Config>
      </orawsp:bindings>
   </orasp:http-oauth2-security>
   <oralgp:Logging orawsp:Silent="true" orawsp:name="Log Message2"
orawsp:Enforced="false" orawsp:category="security/logging">
      <oralgp:msg-log>
         <oralgp:request>all</oralgp:request>
         <oralgp:response>all</oralgp:response>
         <oralgp:fault>all</oralgp:fault>
      </oralgp:msg-log>
      <orawsp:bindings>
         <orawsp:Config orawsp:name="Log Message2_properties">
            <orawsp:PropertySet orawsp:name="standard-security-properties">
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="reference.priority"/>
            </orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:http-oauth2-security>
```

11.5.6.1 Settings

See Table C–45.

11.5.6.2 Configuration Properties

See Table C–8.

11.5.6.3 How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40.

11.5.7 oracle/http_oauth2_token_identity_switch_over_ssl_client_policy

This policy is similar to the policy oracle/ http_oauth2_token_over_ssl_client_policy, with the `subject.precedence` property set to false by default.

This policy includes the OAuth2 access token in the HTTP header.) The access token is obtained from the Mobile and Social OAuth2 Server.) It also verifies that the outbound transport protocol is HTTPS. If a non-HTTPS transport protocol is used, the request is refused.

This policy performs dynamic identity switching by propagating a different identity than the one based on the authenticated subject. This policy can be attached to any HTTP-based SOAP or REST client.

See "Prerequisites for Using the Oracle WSM OAuth2 Policies" on page 10-76.

You can override the following properties when you attach the policy:

- For OAuth2 token request:
  - scope
  - authz.code (Not used in this release.)
  - redirect.uri (Not used in this release.)
- For local token creation:

  - subject.precedence
  - csf.map
  - csf-key
  - oauth2.client.csf.key
  - federated.client.token
  - user.attributes
  - issuer.name
  - oracle.oauth2.service
  - user.roles.include
  - keystore.sig.csf.key
  - propagate.identity.context
  - user.tenant.name
  - include.certificate
- General:
  - audience.uri
  - reference.priority
  - time.in.millis

You must use WLST or edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_oauth2_token_over_ssl_client_template" on page C-69 for information about the assertion attributes that you can configure.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

You attach this policy and the oracle/oauth2_config_client_policy policy to the client application. The `token.uri` property of the required oracle/oauth2_config_client_policy policy specifies the OAuth2 server.

You also attach any of the following Oracle WSM JWT service policies to the web service. The Oracle WSM server-side agent validates the AT.

- oracle/http_jwt_token_over_ssl_service_policy
- oracle/multi_token_over_ssl_rest_service_policy (REST)
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy (SOAP)

`subject.precedence` is set to `false` to allow for the use of a client-specified username rather than the authenticated subject. The user name is obtained only from the username property of the csf-key.

If `subject.precedence` is set to false and `csf-key` and user name are configured, the web service client application must have the `oracle.wsm.security.WSIdentityPermission` permission. That is, applications from which Oracle WSM accepts the externally-supplied identity must have the `WSIdentityPermission` permission. This is to avoid potentially rogue applications from providing an identity to Oracle WSM. See granting `WSIdentityPermission` permission, as described in "Set the WSIdentityPermission Permission" on page 10-80.

By default, the oracle/http_oauth2_token_identity_switch_over_ssl_client_policy assertion content is defined as follows:

```
<orasp:http-oauth2-security
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="Http OAuth2 Over SSL ">
<orasp:auth-header orasp:is-encrypted="false" orasp:is-signed="false"
 orasp:mechanism="oauth2"/>
<orasp:require-tls orasp:algorithm-suite="Basic128"
 orasp:include-timestamp="false" orasp:mutual-auth="false"/>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
 orawsp:name="HttpOAuth2OverSSLConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="subject.precedence">
                  <orawsp:Value>false</orawsp:Value>
               </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="csf.map"/>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="csf-key">
                  <orawsp:Value/>
               </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="oauth2.client.csf.key">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
                </orawsp:Property>
                 <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="federated.client.token">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
                 </orawsp:Property>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="scope">
                    <orawsp:Value/>
                </orawsp:Property>
orawsp:Property orawsp:type="string" orawsp:contentType="optional"
 orawsp:name="authz.code">
                    <orawsp:Value/>
                 </orawsp:Property>
orawsp:Property orawsp:type="string" orawsp:contentType="optional"
 orawsp:name="redirect.uri">
                    <orawsp:Value/>
                </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="user.attributes">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="issuer.name">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>www.oracle.com</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="oracle.oauth2.service">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="user.roles.include">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="reference.priority">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:name="propagate.identity.context"
orawsp:type="string" orawsp:contentType="optional">
                  <orawsp:Value></orawsp:Value>
                </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="user.tenant.name">
                   <orawsp:Value/>
               </orawsp:Property>
<orawsp:Property orawsp:type="string" orawsp:contentType="optional"
 orawsp:name="audience.uri">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="include.certificate">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="time.in.millis">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
      </orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:http-oauth2-security>
```

11.5.7.1 Settings

See Table C–45.

11.5.7.2 Configuration Properties

See Table C–8.

11.5.7.3 How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40.

11.5.8 oracle/http_oauth2_token_opc_oauth2_over_ssl_client_policy

This policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the OAuth2 Server in the Oracle Cloud.

The property `oracle.oauth2.service` is set to true by default, which ensures that the client ID is used as the issuer for the user and client JWT tokens for the OAuth2 server. If `scope` is empty (the default), Oracle WSM automatically gets the service URL and uses the address:port portion as the scope.

The policy verifies that the outbound transport protocol is HTTPS. If a non-HTTPS transport protocol is used, the request is refused. You can attach this policy to any HTTP-based SOAP or REST client.

See "Prerequisites for Using the Oracle WSM OAuth2 Policies" on page 10-76.

You can override the following properties when you attach the policy:

- For OAuth2 token request:
  - scope
  - authz.code (Not used in this release.)
  - redirect.uri (Not used in this release.)
- For local token creation:
  - subject.precedence
  - csf.map
  - csf-key
  - oauth2.client.csf.key
  - federated.client.token
  - user.attributes
  - issuer.name
  - oracle.oauth2.service
  - user.roles.include
  - keystore.sig.csf.key
  - propagate.identity.context
  - user.tenant.name
  - include.certificate
- General:
  - audience.uri
  - reference.priority
  - time.in.millis

You must use WLST or edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_oauth2_token_over_ssl_client_template" on page C-69 for information about the assertion attributes that you can configure.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

You attach this policy and the oracle/oauth2_config_client_policy to the client application. The required `token.uri` property of the oracle/oauth2_config_client_policy policy specifies the OAuth2 server.

You also attach any of the following Oracle WSM JWT service policies to the web service. The Oracle WSM server-side agent validates the AT.

- oracle/http_jwt_token_over_ssl_service_policy
- oracle/multi_token_over_ssl_rest_service_policy (REST)
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy (SOAP)

By default, the oracle/http_oauth2_token_opc_oauth2_over_ssl_client_policy assertion content is defined as follows:

```
<orasp:http-oauth2-security
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="Http OAuth2 Over SSL ">
<orasp:auth-header orasp:is-encrypted="false" orasp:is-signed="false"
 orasp:mechanism="oauth2"/>
<orasp:require-tls orasp:algorithm-suite="Basic128"
 orasp:include-timestamp="false" orasp:mutual-auth="false"/>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
 orawsp:name="HttpOAuth2OverSSLConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="subject.precedence">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="csf.map"/>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="csf-key">
                  <orawsp:Value/>
               </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="oauth2.client.csf.key">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
                </orawsp:Property>
                 <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="federated.client.token">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
                 </orawsp:Property>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="scope">
                    <orawsp:Value/>
                </orawsp:Property>
 <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
 orawsp:name="authz.code">
                    <orawsp:Value/>
                </orawsp:Property>
   <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
 orawsp:name="redirect.uri">
                    <orawsp:Value/>
                </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="user.attributes">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
```

```
orawsp:name="issuer.name">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="oracle.oauth2.service">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="user.roles.include">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="reference.priority">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:name="propagate.identity.context"
orawsp:type="string" orawsp:contentType="optional">
                  <orawsp:Value></orawsp:Value>
                </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="user.tenant.name">
                   <orawsp:Value/>
               </orawsp:Property>
  <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
 orawsp:name="audience.uri">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="include.certificate">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="time.in.millis">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
      </orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:http-oauth2-security>
```

11.5.8.1 Settings

See Table C–45.

11.5.8.2 Configuration Properties

See Table C–8.

11.5.8.3 How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40.

11.5.9 oracle/http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_policy

This policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the OAuth Server in the Oracle Cloud.

The property `oracle.oauth2.service` is set to true by default, which ensures that the client ID is used as the issuer for the user and client JWT tokens for the OAuth2 server. If `scope` is empty (the default), Oracle WSM automatically gets the service URL and uses the address:port portion as the scope.

It also verifies that the outbound transport protocol is HTTPS. If a non-HTTPS transport protocol is used, the request is refused. This policy can be attached to any HTTP-based SOAP or REST client, invoking the service over SSL.

This policy also performs dynamic identity switching by propagating a different identity than the one based on the authenticated subject.

See "Prerequisites for Using the Oracle WSM OAuth2 Policies" on page 10-76.

You can override the following properties when you attach the policy:

- For OAuth2 token request:
  - scope
  - authz.code (Not used in this release.)
  - redirect.uri (Not used in this release.)
- For local token creation:
  - subject.precedence
  - csf.map
  - csf-key
  - oauth2.client.csf.key
  - federated.client.token
  - user.attributes
  - issuer.name
  - oracle.oauth2.service
  - user.roles.include
  - keystore.sig.csf.key
  - propagate.identity.context
  - user.tenant.name
  - include.certificate
- General:
  - audience.uri
  - reference.priority
  - time.in.millis

You must use WLST or edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_oauth2_token_over_ssl_client_template" on page C-69 for information about the assertion attributes that you can configure.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

You attach this policy and the oracle/oauth2_config_client_policy policy to the client application. The token.uri property of the required oracle/oauth2_config_client_policy policy specifies the OAuth2 server.

You also attach any of the following Oracle WSM JWT service policies to the web service. The Oracle WSM server-side agent validates the AT.

- oracle/http_jwt_token_over_ssl_service_policy
- oracle/multi_token_over_ssl_rest_service_policy (REST)
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy (SOAP)

subject.precedence is set to false to allow for the use of a client-specified username rather than the authenticated subject. The user name is obtained only from the username property of the csf-key.

If subject.precedence is set to false and csf-key and user name are configured, the web service client application must have the oracle.wsm.security.WSIdentityPermission permission. That is, applications from which Oracle WSM accepts the externally-supplied identity must have the WSIdentityPermission permission. This is to avoid potentially rogue applications from providing an identity to Oracle WSM. See granting WSIdentityPermission permission, as described in "Set the WSIdentityPermission Permission" on page 10-80.

By default, the oracle/http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_policy assertion content is defined as follows:

```
<orasp:http-oauth2-security
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="Http OAuth2 Over SSL ">
<orasp:auth-header orasp:is-encrypted="false" orasp:is-signed="false"
 orasp:mechanism="oauth2"/>
<orasp:require-tls orasp:algorithm-suite="Basic128"
 orasp:include-timestamp="false" orasp:mutual-auth="false"/>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
 orawsp:name="HttpOAuth2OverSSLConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="subject.precedence">
                  <orawsp:Value>false</orawsp:Value>
               </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="csf.map"/>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="csf-key">
                   <orawsp:Value/>
               </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="oauth2.client.csf.key">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
                </orawsp:Property>
                 <orawsp:Property orawsp:type="boolean"
```

```
orawsp:contentType="optional" orawsp:name="federated.client.token">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
                 </orawsp:Property>
                 <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="scope">
                    <orawsp:Value/>
                </orawsp:Property>
 <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
 orawsp:name="authz.code">
                    <orawsp:Value/>
                </orawsp:Property>
                <orawsp:Property orawsp:type="string"
 orawsp:contentType="optional" orawsp:name="redirect.uri">
                    <orawsp:Value/>
                </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="user.attributes">
                <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="issuer.name">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="oracle.oauth2.service">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="user.roles.include">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="reference.priority">
                  <orawsp:Value/>
               </orawsp:Property>
               <orawsp:Property orawsp:name="propagate.identity.context"
orawsp:type="string" orawsp:contentType="optional">
                  <orawsp:Value></orawsp:Value>
                </orawsp:Property>
                <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="user.tenant.name">
                   <orawsp:Value/>
               </orawsp:Property>
<orawsp:Property orawsp:type="string" orawsp:contentType="optional"
 orawsp:name="audience.uri">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>NONE</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="include.certificate">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>false</orawsp:DefaultValue>
               </orawsp:Property>
               <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="time.in.millis">
                  <orawsp:Value/>
                  <orawsp:DefaultValue>true</orawsp:DefaultValue>
               </orawsp:Property>
      </orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:http-oauth2-security>
```

11.5.9.1 Settings

See Table C–45.

11.5.9.2 Configuration Properties

See Table C–8.

11.5.9.3 How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40.

11.5.10 oracle/http_saml20_bearer_token_over_ssl_client_policy

This policy includes SAML 2.0 tokens in outbound HTTP request messages. The SAML token with confirmation method *Bearer* is created automatically. By default, the authenticated user from the Subject (user principal) is used to generate the SAML assertion for identity propagation.

You must edit the policy file manually; you cannot edit the policy using Fusion Middleware Control. See "oracle/http_saml20_token_bearer_client_template" on page C-20 for information about the assertion attributes that you can configure.

By default, the oracle/http_saml_bearer_token_over_ssl_service_policy assertion content is defined as follows:

```
<orasp:http-saml20-bearer-security
   xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
   xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
   orawsp:Enforced="true" orawsp:Silent="true"
   orawsp:category="security/authentication, security/msg-protection"
   orawsp:name="Http SAML 2.0 Bearer Security Over SSL ">
   <orasp:auth-header orasp:mechanism="saml20-bearer"/>
   <orasp:require-tls orasp:include-timestamp="false"
      orasp:mutual-auth="false"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative"
         orawsp:name="HttpSaml20BearerOverSSLConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="user.attributes" orawsp:type="string"/>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.issuer.name" orawsp:type="string">
               <orawsp:Value>www.oracle.com</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="user.roles.include" orawsp:type="string">
               <orawsp:Value>false</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
```

```
                    orawsp:name="csf-key" orawsp:type="string">
                    <orawsp:Value>basic.credentials</orawsp:Value>
                 </orawsp:Property>
                 <orawsp:Property orawsp:contentType="optional"
                    orawsp:name="subject.precedence" orawsp:type="string">
                    <orawsp:Value>true</orawsp:Value>
                 </orawsp:Property>
                 <orawsp:Property orawsp:contentType="optional"
                    orawsp:name="saml.audience.uri" orawsp:type="string">
                    <orawsp:Value/>
                 </orawsp:Property>
                 <orawsp:Property orawsp:contentType="optional"
                    orawsp:name="keystore.sig.csf.key" orawsp:type="string"/>
                 <orawsp:Property orawsp:contentType="optional"
                    orawsp:name="saml.enveloped.signature.required"
                    orawsp:type="boolean">
                    <orawsp:Value>true</orawsp:Value>
                 </orawsp:Property>
                 <orawsp:Property orawsp:contentType="optional"
                    orawsp:name="reference.priority" orawsp:type="string"/>
                 <orawsp:Property orawsp:name="propagate.identity.context"
                    orawsp:type="string" orawsp:contentType="optional">
                    <orawsp:Value></orawsp:Value>
              </orawsp:PropertySet>
           </orawsp:Config>
        </orawsp:bindings>
     </orasp:http-saml20-bearer-security>
```

11.5.10.1 Settings

See Table C–11.

11.5.10.2 Configuration Properties

See Table C–12.

11.5.11 oracle/http_saml20_bearer_token_over_ssl_service_policy

This policy authenticates users using credentials provided in SAML 2.0 tokens with confirmation method 'Bearer' in the HTTP header.

You must modify the policy file manually; you cannot view, edit, or attach the policy using Fusion Middleware Control. See "oracle/http_saml20_token_bearer_service_template" on page C-23 for information about the assertion attributes that you can configure.

By default, the oracle/http_saml_bearer_token_over_ssl_service_policy policy is defined as follows:

```
<orasp:http-saml20-bearer-security
   xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
   xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
   orawsp:Enforced="true" orawsp:Silent="true"
   orawsp:category="security/authentication"
   orawsp:name="Http SAML 2.0 Bearer Security Over SSL">
   <orasp:auth-header orasp:mechanism="saml20-bearer"/>
   <orasp:require-tls orasp:include-timestamp="false"
      orasp:mutual-auth="false"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative"
      orawsp:name="HttpSaml20BearerOverSSLConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.trusted.issuers" orawsp:type="string">
               <orawsp:Value/>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.enveloped.signature.required"
               orawsp:type="boolean">
               <orawsp:Value>true</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="reference.priority" orawsp:type="string"/>
            <orawsp:Property orawsp:name="propagate.identity.context"
               orawsp:type="string" orawsp:contentType="optional">
               <orawsp:Value></orawsp:Value>
         </orawsp:PropertySet>
      </orawsp:Config>
   </orawsp:bindings>
</orasp:http-saml20-bearer-security>
```

11.5.11.1 Settings

See Table C–11.

11.5.11.2 Configuration Properties

See Table C–13.

11.5.12 oracle/wss_http_token_over_ssl_client_policy

This policy includes credentials in the HTTP header for outbound client requests.

This policy also verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be applied to any HTTP-based endpoint.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following assertion template: oracle/wss_http_token_over_ssl_client_template. See "oracle/wss_http_token_over_ssl_client_template" on page C-70 for more information about the assertion.

11.5.12.1 Settings

See Table C–46.

11.5.12.2 Configuration Properties

See Table C–47.

11.5.12.3 How to Set Up the Web Services Client

You can specify a value for `csf-key` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Adding Keys and User Credentials to the Credential Store" on page 10-19 for information on how to add the key to the credential store.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved. See "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

11.5.12.4 How to Set Up the Web Service Client at Design Time

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

The client must pass the credentials in the HTTP header.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved. See "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

11.5.13 oracle/wss_http_token_over_ssl_service_policy

This policy extracts the credentials in the HTTP header and authenticates users.

This policy verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be applied to any HTTP-based endpoint.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following assertion template: oracle/wss_http_token_over_ssl_service_template. See "oracle/wss_http_token_over_ssl_service_template" on page C-72 for more information about the assertion.

11.5.13.1 Settings

See Table C–46.

11.5.13.2 Configuration Properties

See Table C–48.

11.5.13.3 How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40 if **Allow Mutual Authentication** is checked.

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.5.14 oracle/wss_saml_token_bearer_identity_switch_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method *Bearer* is created automatically. The policy also verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based client.

This policy contains the following policy assertion: oracle/wss_saml_token_bearer_over_ssl_client_template. See "oracle/wss_saml_token_bearer_client_template" on page C-46 for more information about the assertion.

11.5.14.1 Settings

See Table C–32

11.5.14.2 Configuration Properties

See Table C–33

11.5.14.3 How to Set Up the Web Service Client

You attach the wss_saml_token_bearer_identity_switch_client_policy to any SOAP-based Web service client.

Override the configuration properties defined in Table C–33, " wss_saml_token_bearer_client_template Configurations". For more information, see "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

`subject.precedence` is set to `false` to allow for the use of a client-specified username rather than the authenticated subject. (If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the username property of the csf-key.) The wss_saml_token_bearer_identity_switch_client_policy requires that an application that the policy is attached to must have the `WSIdentityPermission` permission. That is, applications from which Oracle WSM accepts the externally-supplied identity must have the `WSIdentityPermission` permission. This is to avoid potentially rogue applications from providing an identity to Oracle WSM.

See "Configuring Web Service Clients for Identity Switching" on page 10-78 for information about how to use this policy, including learning "How the Username Is Picked Up by an Identity Switch Policy on the Client Side" on page 10-79 and granting `WSIdentityPermission` permission, as described in "Set the WSIdentityPermission Permission" on page 10-80.

For additional SAML considerations, see "How to Configure SAML Web Service Client at Design Time" on page 10-65.

11.5.14.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

11.5.15 oracle/wss_saml_token_bearer_over_ssl_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method *Bearer* is created automatically.

This policy contains the following assertion template: oracle/wss_saml_token_bearer_over_ssl_client_template. See "oracle/wss_saml_token_bearer_over_ssl_client_template" on page C-75 for more information about the assertion.

11.5.15.1 Settings

See Table C–51

11.5.15.2 Configuration Properties

See Table C–52

11.5.15.3 How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.15.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

11.5.16 oracle/wss_saml_token_bearer_over_ssl_service_policy

This policy authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

This policy contains the following assertion template: oracle/wss_saml_token_bearer_over_ssl_service_template. See "oracle/wss_saml_token_bearer_over_ssl_service_template" on page C-79 for more information about the assertion.

11.5.16.1 Settings

See Table C–51.

11.5.16.2 Configuration Properties

See Table C–53.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.16.3 Configure the Login Module

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

11.5.16.4 How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

To configure SSL, see "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40 if **Require Mutual Authentication** is checked.

11.5.17 oracle/wss_saml20_token_bearer_over_ssl_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method *Bearer* is created automatically.

This policy contains the following assertion template: oracle/wss_saml20_token_bearer_over_ssl_client_template. See "oracle/wss_saml20_token_bearer_over_ssl_client_template" on page C-81 for more information about the assertion.

11.5.17.1 Settings

See Table C–54.

11.5.17.2 Configuration Properties

See Table C–55.

11.5.17.3 How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.17.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

11.5.18 oracle/wss_saml20_token_bearer_over_ssl_service_policy

This policy authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

This policy contains the following assertion template: oracle/wss_saml20_token_bearer_over_ssl_service_template. See "oracle/wss_saml20_token_bearer_over_ssl_service_template" on page C-85 for more information about the assertion.

11.5.18.1 Settings

See Table C–54.

11.5.18.2 Configuration Properties

See Table C–56.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.18.3 Configure the Login Module

Configure the `saml2.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

11.5.18.4 How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

To configure SSL, see "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40 if **Require Mutual Authentication** is checked.

11.5.19 oracle/wss_saml_token_over_ssl_client_policy

This policy enables the authentication of credentials provided via a SAML token within WS-Security SOAP header.

This policy contains the following assertion template: oracle/wss_saml_token_over_ssl_client_template. See "oracle/wss_saml_token_over_ssl_client_template" on page C-86 for more information about the assertion.

11.5.19.1 Settings

See Table C–57.

11.5.19.2 Configuration Properties

See Table C–58.

11.5.19.3 How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.19.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

11.5.20 oracle/wss_saml_token_over_ssl_service_policy

This policy enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header.

This policy contains the following assertion template: oracle/wss_saml_token_over_ssl_service_template. See "oracle/wss_saml_token_over_ssl_service_template" on page C-90 for more information about the assertion.

11.5.20.1 Settings

See Table C–57

11.5.20.2 Configuration Properties

See Table C–59

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.20.3 Configure the Login Module.

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

11.5.20.4 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

To configure SSL, see "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40 if **Require Mutual Authentication** is checked.

11.5.21 oracle/wss_saml20_token_over_ssl_client_policy

This policy enables the authentication of credentials provided via a SAML token within WS-Security SOAP header.

This policy contains the following assertion template: oracle/wss_saml20_token_over_ssl_client_template. See "oracle/wss_saml20_token_over_ssl_client_template" on page C-91 for more information about the assertion.

11.5.21.1 Settings

See Table C–60.

11.5.21.2 Configuration Properties

See Table C–61.

11.5.21.3 How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.21.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

11.5.22 oracle/wss_saml20_token_over_ssl_service_policy

This policy enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header.

This policy contains the following assertion template: oracle/wss_saml_token_over_ssl_service_template. See "oracle/wss_saml20_token_over_ssl_service_template" on page C-95 for more information about the assertion.

11.5.22.1 Settings

See Table C–60

11.5.22.2 Configuration Properties

See Table C–62.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.22.3 Configure the Login Module.

Configure the `saml2.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

11.5.22.4 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

To configure SSL, see "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40 if **Require Mutual Authentication** is checked.

11.5.23 oracle/wss_username_token_over_ssl_client_policy

This policy includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages. The plain text mechanism is supported. The policy also uses SSL for achieving transport layer security.

This policy contains the following assertion template: oracle/wss_username_token_over_ssl_client_template. See "oracle/wss_username_token_over_ssl_client_template" on page C-96 for more information about the assertion.

11.5.23.1 Settings

See Table C–63.

11.5.23.2 Configuration Properties

See Table C–64.

11.5.23.3 How to Set Up the Web Service Client

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

You can specify a value for *csf-key* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Adding Keys and User Credentials to the Credential Store" on page 10-19 for information on how to add the key to the credential store.

If you specify a password type of None on the **Settings** page, you do not need to include a password in the key.

11.5.23.4 How to Set Up the Web Service Client at Design Time

The client must include a WS-Security UsernameToken element (<wsse:UsernameToken/>) in the SOAP request message. The client provides a username and password for authentication.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved. See "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

11.5.24 oracle/wss_username_token_over_ssl_service_policy

This policy uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users. The plain text mechanism is supported.

This policy contains the following assertion template: oracle/wss_username_token_over_ssl_service_template. See "oracle/wss_username_token_over_ssl_service_template" on page C-99 for more information about the assertion.

11.5.24.1 Settings

See Table C–63.

11.5.24.2 Configuration Properties

See Table C–65.

11.5.24.3 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The username and password must exist and be valid.

To configure SSL, see "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40 if **Require Mutual Authentication** is checked.

11.5.25 oracle/wss10_saml_hok_token_with_message_protection_client_policy

This policy provides message-level protection and SAML holder of key based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_saml_hok_token_with_message_integrity_client_template. See "oracle/wss10_saml_hok_token_with_message_protection_service_template" on page C-105 for more information about the assertion.

11.5.25.1 Settings

See Table C–66.

11.5.25.2 Configuration Properties

See Table C–67.

11.5.25.3 How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

Override the `saml.assertion.filename` property to point to the file that has the holder-of-key assertion.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

You can specify a value for `saml.issuer.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `saml.issuer.name` property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

11.5.25.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

Override the saml.assertion.filename property to point to the file that has the holder-of-key assertion. See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

11.5.26 oracle/wss10_saml_hok_token_with_message_protection_service_policy

This policy enforces message-level protection and SAML holder of key based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_saml_hok_token_with_message_integrity_service_template. See "oracle/wss10_saml_hok_token_with_message_protection_service_template" on page C-105 for more information about the assertion.

11.5.26.1 Configure the Login Module

Configure the `saml.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

11.5.26.2 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

> **Note:** A CertificateExpiredException is returned if an expired certificate is present in the keystore, regardless of whether this certificate is being referenced. To resolve this exception, remove the expired certificate from the keystore.

Store the trusted certificate of the SAML authority in the keystore.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.27 oracle/wss10_saml_token_with_message_integrity_client_policy

This policy provides message-level integrity and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_saml_token_with_message_integrity_client_template. See "oracle/wss10_saml20_token_with_message_protection_client_template" on page C-115 for more information about the assertion.

11.5.27.1 Settings

See Table C–72.

11.5.27.2 Configuration Properties

See Table C–73.

11.5.27.3 How to Set Up the Web Service Client

See "Configuring SAML" on page 10-64.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

You can specify a value for `saml.issuer.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `saml.issuer.name` property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

You can specify a value for `user.roles.include` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for keystore.sig.csf.key and keystore.enc.csf.key on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for propagate.identity.context on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The propagate.identity.context property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.27.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

Include a WS-Security Header Element (<saml:Assertion>) that inserts a SAML token in the outbound SOAP message. The confirmation type is always *sender-vouches*.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

11.5.28 oracle/wss10_saml_token_with_message_integrity_service_policy

This policy enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-113 for more information about the assertion.

11.5.28.1 Settings

See Table C–72.

11.5.28.2 Configuration Properties

See Table C–74.

You also have the option to override the keystore.sig.csf.key and keystore.enc.csf.key server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

You can specify a value for propagate.identity.context on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The propagate.identity.context property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.28.3 Configure the Login Module

Configure the `saml.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.28.4 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

11.5.29 oracle/wss10_saml_token_with_message_protection_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-107 for more information about the assertion.

11.5.29.1 Settings

See Table C–69.

11.5.29.2 Configuration Properties

See Table C–70.

11.5.29.3 How to Set Up the Web Service Client

See "Configuring SAML" on page 10-64.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `saml.issuer.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `saml.issuer.name` property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

You can specify a value for `user.roles.include` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.29.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

11.5.30 oracle/wss10_saml_token_with_message_protection_service_policy

This policy enforces message-level protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-113 for more information about the assertion.

11.5.30.1 Settings

See Table C–69.

11.5.30.2 Configuration Properties

See Table C–71.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See

"Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.30.3 Configure the Login Module

Configure the `saml.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

11.5.30.4 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.31 oracle/wss10_saml20_token_with_message_protection_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_saml20_token_with_message_protection_client_template. See "oracle/wss10_saml20_token_with_message_protection_client_template" on page C-115 for more information about the assertion.

11.5.31.1 Settings

See Table C–72.

11.5.31.2 Configuration Properties

See Table C–73.

11.5.31.3 How to Set Up the Web Service Client

See "Configuring SAML" on page 10-64.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `saml.issuer.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `saml.issuer.name` property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

You can specify a value for `user.roles.include` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.31.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

11.5.32 oracle/wss10_saml20_token_with_message_protection_service_policy

This policy enforces message-level protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_sam20l_token_with_message_protection_service_template. See "oracle/wss10_saml20_token_with_ message_protection_service_template" on page C-121 for more information about the assertion.

11.5.32.1 Settings

See Table C–72.

11.5.32.2 Configuration Properties

See Table C–74.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

You can also specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.32.3 Configure the Login Module

Configure the `saml2.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

11.5.32.4 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.33 oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

This policy uses the Subject Key Identifier (ski) reference mechanism for the encryption key in the request, and for both the signature and encryption keys in the response.

This policy contains the following assertion template: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-107 for more information about the assertion.

> **Note:** Due to the import restrictions of some countries, the jurisdiction policy files distributed with the JDK 5.0 software have built-in restrictions on available cryptographic strength.
>
> By default, policies that use the basic192 algorithms and above do not work with the bundled JRE/JDK. To use these algorithms, you need to download the JCE Extension jars (Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0) file from `http://www.oracle.com/technetwork/java/javase/downloads/index-jdk5-jsp-142662.html`.
>
> To use these policy files, you need to replace the following JAR files in `$JAVA_HOME/jre/lib/security` with the corresponding JARs from the JCE Extension:
>
> - US_export_policy.jar
> - local_policy.jar
>
> You should back up your existing JAR files before replacing them.

11.5.33.1 Settings

See Table C–69.

11.5.33.2 Configuration Properties

See Table C–70.

11.5.33.3 How to Set Up the Web Service Client

See "Configuring SAML" on page 10-64.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for keystore.sig.csf.key and keystore.enc.csf.key on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for saml.issuer.name on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The saml.issuer.name property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

You can specify a value for user.roles.include on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for propagate.identity.context on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The propagate.identity.context property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.33.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

11.5.34 oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy

This policy enforces message-level protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses the Subject Key Identifier (ski) reference mechanism for the encryption key in the request, and for both the signature and encryption keys in the response.

This policy contains the following assertion template: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-113 for more information about the assertion.

Note: Due to the import restrictions of some countries, the jurisdiction policy files distributed with the JDK 5.0 software have built-in restrictions on available cryptographic strength.

By default, policies that use the basic192 algorithms and above do not work with the bundled JRE/JDK. To use these algorithms, you need to download the JCE Extension jars (Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0) file from `http://www.oracle.com/technetwork/java/javase/downloads/index-jdk5-jsp-142662.html`.

To use these policy files, you need to replace the following JAR files in `$JAVA_HOME/jre/lib/security` with the corresponding JARs from the JCE Extension:

- US_export_policy.jar
- local_policy.jar

You should back up your existing JAR files before replacing them.

11.5.34.1 Settings

See Table C–69.

11.5.34.2 Configuration Properties

See Table C–71.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.34.3 Configure the Login Module

Configure the `saml.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60 for more information.

11.5.34.4 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore. When using the ski reference mechanism, use OpenSSL or another such utility to create the certificate.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.35 oracle/wss10_username_id_propagation_with_msg_protection_client_policy

This policy provides message-level protection (that is, integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_username_id_propagation_with_msg_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-123 for more information about the assertion.

11.5.35.1 Settings

See Table C–75.

11.5.35.2 Configuration Properties

See Table C–76.

11.5.35.3 How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

11.5.35.4 How to Set Up the Web Service Client at Design Time

The client must include a WS-Security UsernameToken element (<wsse:UsernameToken/>) in the SOAP request message. The client provides a username and password for authentication.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

Configure the policy assertion for message signing, message encryption, or both.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

11.5.36 oracle/wss10_username_id_propagation_with_msg_protection_service_policy

This policy enforces message level protection (that is, integrity and confidentiality) and identity propagation for inbound SOAP requests using mechanisms described in WS-Security 1.0.

This policy contains the following assertion template: oracle/wss10_username_id_propagation_with_msg_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-127 for more information about the assertion.

11.5.36.1 Settings

See Table C–76.

11.5.36.2 Configuration Properties

See Table C–78. You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.36.3 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

How to Set Up Oracle Platform Security Services (OPSS

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.37 oracle/wss10_username_token_with_message_protection_client_policy

This policy provides message-level protection (message integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-123 for more information about the assertion.

11.5.37.1 Settings

See Table C–75.

11.5.37.2 Configuration Properties

See Table C–76.

11.5.37.3 How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `csf-key` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Adding Keys and User Credentials to the Credential Store" on page 10-19 for information on how to add the key to the credential store.

11.5.37.4 How to Set Up the Web Service Client at Design Time

Configure the policy assertion for message signing, message encryption, or both.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

11.5.38 oracle/wss10_username_token_with_message_protection_service_policy

This policy enforces message-level protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_username_token_with_message_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-127 for more information about the assertion.

11.5.38.1 Settings

See Table C–75.

11.5.38.2 Configuration Properties

See Table C–77. You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.38.3 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.39 oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy

This policy provides message-level protection (message integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses the Subject Key Identifier (ski) reference mechanism for the encryption key in the request, and for both the signature and encryption keys in the response.

This policy contains the following assertion template: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-123 for more information about the assertion.

> **Note:** Due to the import restrictions of some countries, the jurisdiction policy files distributed with the JDK 5.0 software have built-in restrictions on available cryptographic strength.
>
> By default, policies that use the basic192 algorithms and above do not work with the bundled JRE/JDK. To use these algorithms, you need to download the JCE Extension jars (Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0) file from `http://www.oracle.com/technetwork/java/javase/downloads/index-jdk5-jsp-142662.html`.
>
> To use these policy files, you need to replace the following JAR files in `$JAVA_HOME/jre/lib/security` with the corresponding JARs from the JCE Extension:
>
> - US_export_policy.jar
> - local_policy.jar
>
> You should back up your existing JAR files before replacing them.

11.5.39.1 Settings

See Table C–75.

11.5.39.2 Configuration Properties

See Table C–76.

11.5.39.3 How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for keystore.sig.csf.key and keystore.enc.csf.key on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for csf-key on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Adding Keys and User Credentials to the Credential Store" on page 10-19 for information on how to add the key to the credential store.

11.5.39.4 How to Set Up the Web Service Client at Design Time

Configure the policy assertion for message signing, message encryption, or both.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

11.5.40 oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy

This policy enforces message-level protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses the Subject Key Identifier (ski) reference mechanism for the encryption key in the request, and for both the signature and encryption keys in the response.

This policy contains the following assertion template: oracle/wss10_username_token_with_message_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-127 for more information about the assertion.

Note: Due to the import restrictions of some countries, the jurisdiction policy files distributed with the JDK 5.0 software have built-in restrictions on available cryptographic strength.

By default, policies that use the basic192 algorithms and above do not work with the bundled JRE/JDK. To use these algorithms, you need to download the JCE Extension jars (Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0) file from `http://www.oracle.com/technetwork/java/javase/downloads/index-jdk5-jsp-142662.html`.

To use these policy files, you need to replace the following JAR files in `$JAVA_HOME/jre/lib/security` with the corresponding JARs from the JCE Extension:

- US_export_policy.jar
- local_policy.jar

You should back up your existing JAR files before replacing them.

11.5.40.1 Settings

See Table C–75.

11.5.40.2 Configuration Properties

See Table C–77. You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.40.3 How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore. When using the ski reference mechanism, use OpenSSL or another such utility to create the certificate.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.41 oracle/wss10_x509_token_with_message_protection_client_policy

This policy provides message-level protection and certificate credential population for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_x509_token_with_message_protection_client_template. See "oracle/wss10_x509_token_with_message_protection_client_template" on page C-129 for more information about the assertion.

11.5.41.1 Settings

See Table C–78.

11.5.41.2 Configuration Properties

See Table C–79.

11.5.41.3 How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

11.5.41.4 How to Set Up the Web Service Client at Design Time

The Web service client needs to provide valid X.509 authentication credentials in the SOAP message through the WS-Security binary security token.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–3 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 11–4 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

11.5.42 oracle/wss10_x509_token_with_message_protection_service_policy

This policy enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following assertion template: oracle/wss10_x509_token_with_message_protection_service_template. See "oracle/wss10_x509_token_with_message_protection_service_template" on page C-132 for more information about the assertion.

11.5.42.1 Settings

See Table C–78.

11.5.42.2 Attributes You Can Configure

See Table C–80. You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.42.3 How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

How to Set Up WebLogic Server

You need to configure an Authentication provider, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.5.43 oracle/wss11_kerberos_token_with_message_protection_client_policy

This policy includes a Kerberos token in the WS-Security header, and uses Kerberos keys to guarantee message integrity and confidentiality, in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

This policy contains the following assertion template: oracle/wss11_kerberos_token_with_message_protection_client_template. See "oracle/wss11_kerberos_token_with_message_protection_client_template" on page C-134 for more information about the assertion.

11.5.43.1 Settings

See Table C–81.

11.5.43.2 Configuration Properties

See Table C–82.

11.5.43.3 How to Set up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

Also see "Using Kerberos Tokens" on page 10-85.

11.5.43.4 How to Set Up the Web Service Client at Design Time

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16.

Also see "Using Kerberos Tokens" on page 10-85.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–5 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.1 standards. In this example, the body element is encrypted.

11.5.44 oracle/wss11_kerberos_token_with_message_protection_service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

This policy contains the following assertion template: oracle/wss11_kerberos_token_with_message_protection_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-137 for more information about the assertion.

11.5.44.1 Settings

See Table C–81.

11.5.44.2 Configuration Properties

You have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.44.3 Configure the Login Module

Configure the `krb5.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60.

11.5.44.4 How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Configure Kerberos, as described in "Using Kerberos Tokens" on page 10-85.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.5.45 oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy

This policy includes a Kerberos token in the WS-Security header, and uses Kerberos keys to guarantee message integrity and confidentiality, in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

The policy uses Basic128 as the algorithm suite.

This policy contains the following assertion template: oracle/wss11_kerberos_token_with_message_protection_client_template. See "oracle/wss11_kerberos_token_with_message_protection_client_template" on page C-134 for more information about the assertion.

11.5.45.1 Settings

See Table C–81.

11.5.45.2 Configuration Properties

See Table C–82.

11.5.45.3 How to Set up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

Also see "Using Kerberos Tokens" on page 10-85.

11.5.45.4 How to Set Up the Web Service Client at Design Time

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16.

Also see "Using Kerberos Tokens" on page 10-85.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–5 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.1 standards. In this example, the body element is encrypted.

11.5.46 oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

The policy uses Basic128 as the algorithm suite.

This policy contains the following assertion template: oracle/wss11_kerberos_token_with_message_protection_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-137 for more information about the assertion.

11.5.46.1 Settings

See Table C–81.

11.5.46.2 Configuration Properties

You have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.46.3 Configure the Login Module

Configure the `krb5.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60.

11.5.46.4 How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Configure Kerberos, as described in "Using Kerberos Tokens" on page 10-85.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.5.47 oracle/wss11_saml_token_with_message_protection_client_policy

This policy enables message level protection and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

This policy contains the following assertion template: oracle/wss11_saml_token_with_message_protection_client_template. See "oracle/wss11_saml_token_with_message_protection_client_template" on page C-138 for more information about the assertion.

11.5.47.1 Settings

See Table C–83.

11.5.47.2 Configuration Properties

See Table C–84.

11.5.47.3 How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `saml.issuer.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `saml.issuer.name` property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

You can specify a value for `user.roles.include` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.47.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–5 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.1 standards. In this example, the body element is encrypted.

11.5.48 oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy

This policy enables identity switching. Identity switching means that the policy propagates a different identity than the one based on the authenticated Subject. Instead of using the username from the Subject, this policy allows you to set a new user name when sending the SAML Web service request.

This policy enables message level protection and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

This policy contains the following assertion template: oracle/wss11_saml_token_with_message_protection_client_template. See "oracle/wss11_saml_token_with_message_protection_client_template" on page C-138 for more information about the assertion.

11.5.48.1 Settings

See Table C–83.

11.5.48.2 Configuration Properties

See Table C–84.

11.5.48.3 How to Set Up the Web Service Client

You attach the wss11_saml_token_identity_switch_with_message_protection_client_policy to a Web service client of any type.

Override the configuration properties defined in Table C–33, " wss_saml_token_bearer_client_template Configurations". For more information, see "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

`subject.precedence` is set to `false` to allow for the use of a client-specified username rather than the authenticated subject. (If subject.precedence is false, the user name to create the SAML assertion is obtained only from the username property of the csf-key.) The wss11_saml_token_identity_switch_with_message_protection_client_policy policy requires that an application to which the policy is attached must have the `WSIdentityPermission` permission. That is, applications from which Oracle WSM accepts the externally-supplied identity must have the `WSIdentityPermission` permission. This is to avoid potentially rogue applications from providing an identity to Oracle WSM.

See "Configuring Web Service Clients for Identity Switching" on page 10-78 for information about how to use this policy, including learning "How the Username Is Picked Up by an Identity Switch Policy on the Client Side" on page 10-79 and granting

WSIdentityPermission permission, as described in "Set the WSIdentityPermission Permission" on page 10-80.

See "How to Configure SAML Web Service Client at Design Time" on page 10-65 for additional SAML considerations.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `saml.issuer.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `saml.issuer.name` property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

You can specify a value for `saml.audience.uri` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `user.roles.include` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

11.5.48.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–5 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.1 standards. In this example, the body element is encrypted.

11.5.49 oracle/wss11_saml_token_with_message_protection_service_policy

This policy enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following assertion template: oracle/wss11_saml_token_with_message_protection_service_template. See "oracle/wss11_saml_token_with_message_protection_service_template" on page C-143 for more information about the assertion.

11.5.49.1 Settings

See Table C–83.

11.5.49.2 Configuration Properties

See Table C–85.

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.49.3 Configure the Login Module

Configure the `saml.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60.

11.5.49.4 How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

11.5.50 oracle/wss11_saml20_token_with_message_protection_client_policy

This policy enables message level protection and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

This policy contains the following assertion template: oracle/wss11_saml20_token_with_message_protection_client_template. See "oracle/wss11_saml20_token_with_message_protection_client_template" on page C-145 for more information about the assertion.

11.5.50.1 Settings

See Table C–86.

11.5.50.2 Configuration Properties

See Table C–87.

11.5.50.3 How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `saml.issuer.name` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `saml.issuer.name` property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 10-67 for additional considerations.

You can specify a value for `user.roles.include` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.50.4 How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 10-65.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–5 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.1 standards. In this example, the body element is encrypted.

11.5.51 oracle/wss11_saml20_token_with_message_protection_service_policy

This policy enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following assertion template: oracle/wss11_saml20_token_with_message_protection_service_template. See "oracle/wss11_saml20_token_with_message_protection_service_template" on page C-151 for more information about the assertion.

11.5.51.1 Settings

See Table C–86.

11.5.51.2 Configuration Properties

See Table C–88.

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

You can specify a value for `propagate.identity.context` on the **Configurations** page, or override it using the **Security Configuration Details** control when you attach the policy. The `propagate.identity.context` property defaults to a value of blank. See "Propagating Identity Context with Oracle WSM" on page 10-81 for additional considerations.

11.5.51.3 Configure the Login Module

Configure the `saml2.loginmodule` login module. See "Configuring the SAML and Kerberos Login Modules" on page 10-60.

11.5.51.4 How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 10-64.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

11.5.52 oracle/wss11_username_token_with_message_protection_client_policy

This policy provides message-level protection and authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following assertion template: oracle/wss11_username_token_with_message_protection_client_template. See "oracle/wss11_username_token_with_message_protection_client_template" on page C-153 for more information about the assertion.

11.5.52.1 Settings

See Table C–89.

11.5.52.2 Configuration Properties

See Table C–90.

11.5.52.3 How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.enc.csf.key` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

11.5.52.4 How to Set Up the Web Service Client at Design Time

This policy uses symmetric key technology, which is an encryption method that uses the same shared key to encrypt and decrypt data. The symmetric key is used to sign the message.

Configure the policy assertion for message signing, message encryption, or both.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Example 11–5 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.1 standards. In this example, the body element is encrypted.

11.5.53 oracle/wss11_username_token_with_message_protection_service_policy

This policy enforces message-level protection (that is, message integrity and message confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following assertion template: oracle/wss11_username_token_with_message_protection_service_template. See "oracle/wss11_username_token_with_message_protection_service_template" on page C-157 for more information about the assertion.

11.5.53.1 Settings

See Table C–89.

11.5.53.2 Configuration Properties

See Table C–91. You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.53.3 How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.5.54 oracle/wss11_x509_token_with_message_protection_client_policy

This policy provides message-level protection and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following assertion template: oracle/wss11_x509_token_with_message_protection_client_template. See "oracle/wss11_x509_token_with_message_ protection_client_template" on page C-159 for more information about the assertion.

11.5.54.1 Settings

See Table C–92.

11.5.54.2 Configuration Properties

See Table C–93.

11.5.54.3 How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Oracle WSM keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias` on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key` on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

11.5.54.4 How to Set Up the Web Service Client at Design Time

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore" on page 10-16. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

The Web service client needs to provide valid X.509 authentication credentials in the SOAP message through the WS-Security binary security token.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 11–5 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.1 standards. In this example, the body element is encrypted.

11.5.55 oracle/wss11_x509_token_with_message_protection_service_policy

This policy enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following assertion template: oracle/wss11_x509_token_with_message_protection_service_template. See "oracle/wss11_x509_token_with_message_protection_service_template" on page C-162 for more information about the assertion.

11.5.55.1 Settings

See Table C–92.

11.5.55.2 Configuration Properties

See Table C–94. You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.5.55.3 How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19. Use `keystore.enc.csf.key` as the key name.

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

How to Set Up WebLogic Server

You need to configure the Authentication provider, as described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.

11.6 Authorization Policies and Configuration Steps

Frequently, authentication is the first step of determining whether a user should be given access to a Web service. After the user is authenticated, the second step is to verify that the user is authorized to access the Web service. This is accomplished using an authorization policy. You can create an authorization policy using the binding_authorization_template or the component_authorization_template assertion templates.

Policies created with these templates perform role- or permission-based access control (RBAC) and check that the authenticated user has been granted one of the roles or permissions allowed access to the Web service.

Appendix B, "Predefined Policies" summarizes the security policies that enforce authorization, and indicates whether the policy is enforced at the transport layer or SOAP header.

Note: The authorization polices can follow any authentication policy where the subject is established.

You cannot attach both a permitall and denyall policy to the same Web service.

11.6.1 Determining Which Resources to Protect

The authorization policies provide the following properties that you can use to specify which resources you want the policy to protect. Not all of the predefined policies feature all of the properties.

- Constraint Pattern -- Expression that represents the constraints against which authorization checks are performed. The constraints expression is specified using the following two `messageContext` properties:

  – `messageContext.authenticationMethod`—Determines the authentication method used to authenticate the user. The only valid value is `SAML_SV`.

  – `messageContext.requestOrigin`—Determines whether the request originated from an internal or external network. This property is valid only when using Oracle HTTP Server and the Oracle HTTP server administrator has added a custom `VIRTUAL_HOST_TYPE` header to the request. For details about adding this header to a request, see "Configuring Oracle HTTP Server to Specify Request Origin" on page 11-108.

  Note the following:

  – The Constraint Pattern properties and their values are case sensitive.

  – The constraint expression uses the following standard supported operators: `==`, `!=`, `&&`, `||` and `!`.

  In the following example, the role-based authorization assertion will be executed only if the current message does *not* contain a SAML_SV token OR the request origin is not internal.

  ```
  ${!(messageContext.authenticationMethod =='SAML_SV'||
  messageContext.requestOrigin == 'internal')}
  ```

  > **Note:** This property is valid for authorization policies based on the binding_authorization_template only. For policies based on other authorization assertion templates, this property is reserved for future use.

- Action Pattern -- The Web service operation for which permission-based checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. * means all Web service operations.

  The valid values for Action Pattern are determined by the Web service methods. For example, if the Web service method is *validate(amountAvailable)*, enter the Action Pattern as *validate*.

- Resource Pattern -- The name of the resource for which permission-based checks are performed. This field accepts wildcards, and the default is * for all resources in the Web services protected by the policy.

  By convention you enter the Resource Pattern as (namespace of Web service + Web service name).

  For example, if the namespace of the Web service is *http://project11* and the Web service name is *CreditValidation*, you would enter the Resource Name as *http://project11/CreditValidation*.

  If you specify a specific Resource Pattern, the policy is enforced only for those Web services that match the criteria. That is, entering a specific Resource Pattern limits the scope of the authorization policy. This condition also applies if you have bulk-attached this authorization policy to multiple subjects. The default of * protects all resources (namespace of Web service + Web service name) of the bulk-attached Web services.

- Permission Check Class -- By default, it is *oracle.wsm.security.WSFunctionPermission*. The class must be in the classpath.
- Authorization Setting -- Possible values are Permit All, Deny All, and Selected Roles. If you choose Selected Roles, you must then select from the enterprise (Global) roles defined in WebLogic Server, which may include the following:
  - AdminChannelUser
  - Anonymous
  - AppTester
  - CrossDomainConnector
  - Deployer
  - Monitor
  - Operator
  - OracleSystemRole

11.6.2 How Authorization Permissions Are Determined

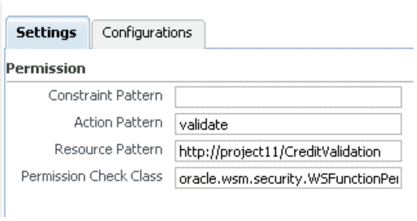
Conceptually, determining whether an authenticated subject is authorized to access a particular resource protected by a Web service policy has two parts that work in tandem.

- The **Resource Pattern**, and **Action Pattern** parameters on the Policy Settings page for the policy determine what resources are being protected by the policy, as shown in Figure 11–2. (You can also override the Resource Pattern and Action Pattern properties, as described in "Configuring Server-Side Override Properties for Authorization Policies" on page 8-28.)

  You have the option to change the *Permission Check Class* configuration property for the policy, which identifies the permission class as per JAAS standards. The permission class must be available in the application or server classpath.

  The custom permission class must extend the abstract *Permission* class and implement the *Serializable* interface. See the Javadoc at `http://java.sun.com/j2se/1.5.0/docs/api/java/security/Permission.html`. The default is *oracle.wsm.security.WSFunctionPermission*.

Figure 11–2 The Permission Settings for a Policy



Constraint Pattern

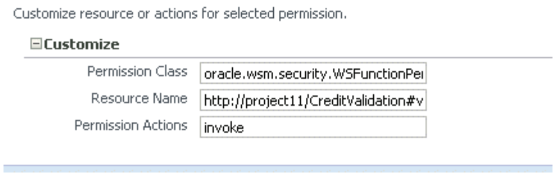
Action Pattern: validate

Resource Pattern: http://project11/CreditValidation

Permission Check Class: oracle.wsm.security.WSFunctionPer

- The OPSS Application Policies page specifies whether the authenticated subject has invoke access to the **Resource Name** listed there, as shown in Figure 11–3.

Figure 11–3 Adding a Permission on the OPSS Create Application Grant Page



Customize resource or actions for selected permission.

⊟Customize

Permission Class: oracle.wsm.security.WSFunctionPer

Resource Name: http://project11/CreditValidation#v

Permission Actions: invoke

OPSS uses the Policy Settings page for the Web service to determine which resources require an authorization check. Then, access to the resource is allowed if the authenticated subject has been granted *WSFunctionPermission* (or other permission) for that resource via OPSS.

> **Note:** If you changed the *Permission Check Class* configuration property for the policy to a custom class, use the custom class here as well.

Consider further the example shown in Figure 11–2 and Figure 11–3.

On the Policy Settings page, assume that you specify the following to protect the *validate* method of the *http://project11/CreditValidation* Web service:

```
Action pattern:        validate
Resource pattern:       http://project11/CreditValidation
Permission Check Class  oracle.wsm.security.WSFunctionPermission
```

Then, on the OPSS Application Policies page, you would use *http://project11/CreditValidation#validate* for the **Resource Name** to specify that the authenticated subject has permission to invoke this resource:

```
Permission Class: oracle.wsm.security.WSFunctionPermission
Resource Name:    http://project11/CreditValidation#validate
Permissions Action:  invoke
```

You can grant the *WSFunctionPermission* permission to a user, a group, or an application role. If you grant *WSFunctionPermission* to a user or group it will apply to all applications that are deployed in the domain.

11.6.2.1 OPSS Resource Name Can Include Operation Name

In previous releases of Fusion Middleware Control, the **Resource Name** on the OPSS Application Policies page was determined by *name-space-of-webservice/ServiceName*. For example, if the name space of a Web service was *http://project1/* and the service name was *CreditValidation*, the **Resource Name** would have been *http://project1/CreditValidation*. You could also use an asterisk (*) wildcard for providing permission to all the actions or all resources.

In this release, the resource target of the *WSFunctionPermission* is enhanced to include the actual Web service operation name. The syntax for the **Resource Name** is now *name-space-of-webservice/servicename#[operation name]*. (For a component it is *compositename/componentname#[operation name]*.)]

You must now include at least the *name-space-of-webservice/service name*. That is, you can no longer use an asterisk (*) wildcard for providing permission to all the actions or all resources.

Instead, to specify all operations for a Web service, simply leave the operation name blank. For example, *name-space-of-webservice/servicename#*

Permission Action is always *invoke*.

11.6.3 oracle/binding_authorization_denyall_policy

This policy provides a simple role-based authorization policy based on the authenticated subject.

This policy denies all users with any role.

This policy should follow an authentication policy where the subject is established and can be attached to any SOAP-based endpoint.

You must have already configured a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

This policy contains the following assertion template: oracle/binding_authorization_template

See "oracle/binding_authorization_template" on page C-183 for more information about the assertion.

11.6.3.1 Settings

See Table C–109.

To add roles:

1. Click **Selected Roles**.
2. Click **Add**.
3. To add roles, click the check box next to each role you want to add in the Roles Available column and click **Move**. To add all roles, click **Move All**.

   To remove roles, click the check box next to each role you want to remove in the Roles Selected to Add column, and click **Remove**. To remove all roles, click **Remove All**.

   To search for roles, enter a search string in the Role Name search box and click the go arrow. The Roles Available column is updated to include only those roles that match the search string.
4. Click **OK**.

To delete roles:

1. Select the role that you want to delete in the Selected Roles list.
2. Click **Delete**.

11.6.3.2 Configuration Properties

None defined.

11.6.3.3 How to Set Up Oracle Platform Security Services (OPSS)

If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. You use the WebLogic Server Administration Console to grant a role to a user or group, as described in the *Oracle WebLogic Server Administration Console Help*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

11.6.4 oracle/binding_authorization_permitall_policy

This policy provides a simple role-based authorization policy based on the authenticated subject.

This policy permits all users with any roles.

This policy should follow an authentication policy where the subject is established and can be attached to any SOAP-based endpoint.

You must have already configured a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

This policy contains the following assertion template: oracle/binding_authorization_template. See "oracle/binding_authorization_template" on page C-183 for more information about the assertion.

11.6.4.1 Settings

See Table C–109.

To add roles:

1. Click **Selected Roles**.
2. Click **Add**.
3. To add roles, click the check box next to each role you want to add in the Roles Available column and click **Move**. To add all roles, click **Move All**.

   To remove roles, click the check box next to each role you want to remove in the Roles Selected to Add column, and click **Remove**. To remove all roles, click **Remove All**.

   To search for roles, enter a search string in the Role Name search box and click the go arrow. The Roles Available column is updated to include only those roles that match the search string.
4. Click **OK**.

To delete roles:

1. Select the role that you want to delete in the Selected Roles list.
2. Click **Delete**.

11.6.4.2 Configuration Properties

None defined.

11.6.4.3 How to Set Up Oracle Platform Security Services (OPSS)

If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. You use the WebLogic Server Administration Console to grant a role to a user or group, as described in the *Oracle WebLogic Server Administration Console Help*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

11.6.5 oracle/binding_permission_authorization_policy

This policy provides a permission-based authorization policy based on the authenticated subject.

This policy ensures that the subject has permission to perform the operation. To do this, the Authorization Policy executor leverages OPSS to check if the authenticated subject has been granted *oracle.wsm.security.WSFunctionPermission* (or whatever permission class is specified in *Permission Check Class*) using the *Resource Pattern* and *Action Pattern* as parameters.

This policy should follow an authentication policy where the subject is established and can be attached to any SOAP-based endpoint.

This policy contains the following assertion template: oracle/binding_permission_authorization_template. See "oracle/binding_permission_authorization_template" on page C-185 for more information about the assertion.

11.6.5.1 Settings

See Table C–110.

11.6.5.2 Attributes You Can Configure

You have the option to change the *permission_class* configuration property for the policy, which identifies the permission class as per JAAS standards. The permission class must be available in the application or server classpath.

The custom permission class must extend the abstract *Permission* class and implement the *Serializable* interface. See the Javadoc at `http://java.sun.com/j2se/1.5.0/docs/api/java/security/Permission.html`.

The default is *oracle.wsm.security.WSFunctionPermission*.

11.6.5.3 How to Set Up Oracle Platform Security Services (OPSS)

Use Fusion Middleware Control to grant the *WSFunctionPermission* (or other) permission to the user, group, or application that will attempt to authenticate to the Web service.

You have the option to change the *permission_class* configuration property for the policy, which identifies the permission class as per JAAS standards. The class must be available in the server classpath. The default is *oracle.wsm.security.WSFunctionPermission*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

11.6.6 oracle/component_authorization_denyall_policy

This policy provides a simple role-based authorization policy based on the authenticated subject.

This policy denies all users with any roles.

You must have already configured a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

This policy should follow an authentication policy where the subject is established and can be attached to any SCA-based endpoint.

This policy contains the following assertion template: oracle/component_authorization_template. See "oracle/component_authorization_template" on page C-186 for more information about the assertion.

11.6.6.1 Settings

See Table C–112.

To add roles:

1. Click **Add**.
2. To add roles, click the check box next to each role you want to add in the Roles Available column and click **Move**. To add all roles, click **Move All**.

   To remove roles, click the check box next to each role you want to remove in the Roles Selected to Add column, and click **Remove**. To remove all roles, click **Remove All**.

   To search for roles, enter a search string in the Role Name search box and click the go arrow. The Roles Available column is updated to include only those roles that match the search string.
3. Click **OK**.

To delete roles:

1. Select the role that you want to delete in the Selected Roles list.
2. Click **Delete**.

11.6.6.2 Configuration Properties

None defined.

11.6.6.3 How to Set Up Oracle Platform Security Services (OPSS)

If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. You use the WebLogic Server Administration Console to grant a role to a user or group, as described in the *Oracle WebLogic Server Administration Console Help*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

11.6.7 oracle/component_authorization_permitall_policy

This policy provides a simple role-based authorization policy based on the authenticated subject.

This policy permits all users with any roles.

You must have already configured a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

It should follow an authentication policy where the subject is established and can be attached to any SCA-based endpoint.

This policy contains the following assertion template: oracle/component_ authorization_template. See "oracle/component_authorization_template" on page C-186 for more information about the assertion.

11.6.7.1 Settings

See Table C–112.

To add roles:

1. Click **Add**.
2. To add roles, click the check box next to each role you want to add in the Roles Available column and click **Move**. To add all roles, click **Move All**.

   To remove roles, click the check box next to each role you want to remove in the Roles Selected to Add column, and click **Remove**. To remove all roles, click **Remove All**.

   To search for roles, enter a search string in the Role Name search box and click the go arrow. The Roles Available column is updated to include only those roles that match the search string.
3. Click **OK**.

To delete roles:

1. Select the role that you want to delete in the Selected Roles list.
2. Click **Delete**.

11.6.7.2 Configuration Properties

None defined.

11.6.7.3 How to Set Up Oracle Platform Security Services (OPSS)

If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. You use the WebLogic Server Administration Console to grant a role to a user or group, as described in the *Oracle WebLogic Server Administration Console Help*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

11.6.8 oracle/component_permission_authorization_policy

This policy provides a permission-based authorization policy based on the authenticated subject.

This policy ensures that the subject has permission to perform the operation. To do this, the Authorization Policy executor leverages OPSS to check if the authenticated subject has been granted *oracle.wsm.security.WSFunctionPermission* (or whatever permission class is specified in *Permission Check Class*) using the *Resource Pattern* and *Action Pattern* as parameters. *Resource Pattern* and *Action Pattern* are used to identify if the authorization assertion is to be enforced for this particular request. Access is allowed if the authenticated subject has been granted *WSFunctionPermission*.

You can grant the *WSFunctionPermission* permission to a user, a group, or an application role. If you grant *WSFunctionPermission* to a user or group it will apply to all applications that are deployed in the domain.

This policy should follow an authentication policy where the subject is established and can be attached to any SCA-based endpoint.

This policy contains the following assertion template: oracle/component_permission_authorization_template. See "oracle/component_permission_authorization_template" on page C-187 for more information about the assertion.

11.6.8.1 Settings

See Table C–113.

11.6.8.2 Configuration Properties

None defined.

11.6.8.3 How to Set Up Oracle Platform Security Services (OPSS)

Use Fusion Middleware Control to grant the *WSFunctionPermission* permission to the user, group, or application that will attempt to authenticate to the Web service.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

11.6.9 oracle/whitelist_authorization_policy

This policy is a special case of role-based authorization policy based on the authenticated subject.

This policy will let requests in only if one of the following conditions is true:

- The authenticated token is SAML Sender Vouches.
- The user is in a particular role (the default is `trustedEnterpriseRole`, that establishes the user as a trusted entity
- The request is coming from within a private network.

This policy can be attached to any SOAP-based endpoint.

This policy contains the following assertion template: oracle/binding_authorization_template.

See "oracle/binding_authorization_template" on page C-183 for more information about the assertion.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

11.6.9.1 Settings

See Table C–109.

To add roles:

1. Click **Add**.
2. To add roles, click the check box next to each role you want to add in the Roles Available column and click **Move**. To add all roles, click **Move All**.

To remove roles, click the check box next to each role you want to remove in the Roles Selected to Add column, and click **Remove**. To remove all roles, click **Remove All**.

To search for roles, enter a search string in the Role Name search box and click the go arrow. The Roles Available column is updated to include only those roles that match the search string.

3. Click **OK**.

To delete roles:

1. Select the role that you want to delete in the Selected Roles list.
2. Click **Delete**.

11.6.9.2 Configuration Properties

None defined.

11.6.9.3 How to Set Up Oracle Platform Security Services (OPSS)

If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. You use the WebLogic Server Administration Console to grant a role to a user or group, as described in the *Oracle WebLogic Server Administration Console Help*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Help*.

11.6.9.4 How to Successfully Invoke Services Using This Policy

To successfully invoke a service that has the whitelist_authorization_policy attached, you must do one of the following:

- If the service accepts SAML sender vouches for authentication (for example, a SAML token service policy is attached to the service), you must attach the corresponding SAML token client policy to the client.
- If the service accepts username/password for authentication (for example, a username token service policy is attached to the service), you must attach the corresponding username token client policy to the client and make sure that the client is in a trusted role as defined in the policy. (By default, the role defined in the predefined policy is `trustedEnterpriseRole`. You need to modify this role in the predefined policy.)
- If the service is invoked using Oracle HTTP Server, and it is configured to indicate that the request came from a private internal network (see "Configuring Oracle HTTP Server to Specify Request Origin" on page 11-108), then a client on the internal network only has to attach the corresponding username token client policy at the client side.

11.6.9.5 Configuring Oracle HTTP Server to Specify Request Origin

The **Constraint Pattern** property setting contains a `requestOrigin` field that specifies whether the request originated from an internal or external network. This property is valid only when using Oracle HTTP Server and the Oracle HTTP server administrator has added a custom `VIRTUAL_HOST_TYPE` header to the request.

To do so, the administrator must modify the `httpd.conf` file as follows:

1. Verify that the module `mod_headers` is loaded.

2. Set the `VIRTUAL_HOST_TYPE` header name in the `RequestHeader`. Valid values are `internal` and `external`. Use the following command syntax:

   ```
   RequestHeader set|append|add|unset header [value [env=[!]variable]]
   ```

   For example, to configure the virtual host for internal requests:

   ```
   <VirtualHost *:7777>
   RequestHeader set VIRTUAL_HOST_TYPE  "internal"
   </VirtualHost>
   ```

   To configure the virtual host for external requests:

   ```
   <VirtualHost *:8888>
   RequestHeader set VIRTUAL_HOST_TYPE "external"
   </VirtualHost>
   ```

   In these examples, all the requests coming from outside of the private network are routed through `virtual host:8888` and all the requests coming from the internal private network are routed through `virtual host:7777`.

   Note that you must also add these ports in the `httpd.conf` file as listen ports so that the applications are available on the ports externally.

3. Restart the Oracle HTTP Server.

11.7 WS-Addressing Policies and Configuration Steps

The Web Services Addressing (WS-Addressing) specification (`http://www.w3.org/TR/ws-addr-core/`) provides transport-neutral mechanisms to address Web services and messages. In particular, the specification defines a number of XML elements used to identify Web service endpoints and to secure end-to-end endpoint identification in messages.

This section describes the predefined WS-Addressing policies.

11.7.1 oracle/wsaddr_policy

This policy causes the platform to check inbound messages for the presence of WS-Addressing headers conforming to the W3C 2005 Final WS-Addressing Policy standard. In addition, it causes the platform to include a WS-Addressing header in outbound SOAP messages.

11.7.1.1 How to Set Up the Web Service Client

No configuration is needed.

11.7.1.2 How to Set Up the Web Service Client at Design Time

Configure WS-Addressing for the Web service client as described in the *Web Services Addressing 1.0 - SOAP Binding* specification (`http://www.w3.org/TR/ws-addr-soap/`).

11.7.1.3 How to Set Up Oracle Platform Security Services (OPSS)

No configuration is needed.

11.8 WS-Trust Policies

This section describes the predefined WS-Trust policies. The predefined policies conform to the WS-Trust 1.3 specification.

11.8.1 oracle/sts_trust_config_service_policy

Use this policy to specify the STS configuration information that is used to invoke the STS for token exchange.

This policy contains the following assertion template: oracle/sts_trust_config_service_template. See "oracle/sts_trust_config_service_template" on page C-166 for more information about the assertion.

11.8.1.1 Policy Assertion

The `oracle/sts_trust_config_service_policy` policy assertion is as follows:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<wsp:Policy
    xmlns:oralgp = "http://schemas.oracle.com/ws/2006/01/loggingpolicy"
    xmlns:orasp = "http://schemas.oracle.com/ws/2006/01/securitypolicy"
    orawsp:description =
"i18n:oracle.wsm.resources.policydescription.PolicyDescriptionBundle_oracle/sts_
trust_config_service_policy_PolyDescKey"
    orawsp:displayName =
"i18n:oracle.wsm.resources.policydescription.PolicyDescriptionBundle_oracle/sts_
trust_config_service_policy_PolyDispNameKey"
    wsu:Id = "sts_trust_config_service_policy"
    orawsp:attachTo = "binding.server"
    orawsp:status = "enabled"
    xmlns:orawsp = "http://schemas.oracle.com/ws/2006/01/policy"
    Name = "oracle/sts_trust_config_service_policy"
    xmlns:wsp = "http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsu =
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xs
d"
    orawsp:category = "security"
    orawsp:local-optimization = "off">

    <orasp:sts-trust-config
        orawsp:Silent = "true"
        orawsp:Enforced = "true"
        orawsp:name = "STS Trust Configuration"
        orawsp:category = "security/sts-config"
        orasp:wsdl-uri = "http://host:port/sts?wsdl"
        orasp:port-uri = "http://host:port/sts-service"
        orasp:soap-version="12">
        <orawsp:bindings>
            <orawsp:Config orawsp:name = "StsTrustConfig" orawsp:configType =
"declarative">
             <orawsp:PropertySet orawsp:name="standard-security-properties">
               <orawsp:Property orawsp:name="role" orawsp:type="string"
orawsp:contentType="constant">
                  <orawsp:Value>ultimateReceiver</orawsp:Value>
               </orawsp:Property>
             </orawsp:PropertySet>
            </orawsp:Config>
        </orawsp:bindings>
    </orasp:sts-trust-config>
</wsp:Policy>
```

11.8.1.2 Settings

You can change the settings shown in Table C–98.

11.8.1.3 Configuration Properties

You can configure the properties shown in Table C–99.

11.8.1.4 How to Set Up the Web Service

See "Setting Up Automatic Policy Configuration for STS" on page 10-104 for the steps to follow.

11.8.2 oracle/sts_trust_config_client_policy

Use this policy to specify the STS client configuration information that is used to invoke the STS for token exchange.

Use this policy only if you are not using Automatic (Client STS) Policy Configuration, as described in "Setting Up Automatic Policy Configuration for STS" on page 10-104

If you attach multiple instances of `oracle/sts_trust_config_client_policy`, no error is generated. However, only one instance is enforced, and you cannot control which instance that is.

This policy contains the following assertion template: oracle/sts_trust_config_template. See "oracle/sts_trust_config_client_template" on page C-165 for more information about the assertion.

11.8.2.1 Policy Assertion

The `oracle/sts_trust_config_client_policy` policy assertion is as follows:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<wsp:Policy
    xmlns:oralgp = "http://schemas.oracle.com/ws/2006/01/loggingpolicy"
    xmlns:orasp = "http://schemas.oracle.com/ws/2006/01/securitypolicy"
    orawsp:description =
 "i18n:oracle.wsm.resources.policydescription.PolicyDescriptionBundle_oracle/sts_
trust_config_client_policy_PolyDescKey"
    orawsp:displayName =
 "i18n:oracle.wsm.resources.policydescription.PolicyDescriptionBundle_oracle/sts_
trust_config_client_policy_PolyDispNameKey"
    wsu:Id = "sts_trust_config_client_policy"
    orawsp:attachTo = "binding.client"
    orawsp:status = "enabled"
    xmlns:orawsp = "http://schemas.oracle.com/ws/2006/01/policy"
    Name = "oracle/sts_trust_config_client_policy"
    xmlns:wsp = "http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsu =
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xs
d"
    orawsp:category = "security"
    orawsp:local-optimization = "off">

    <orasp:sts-trust-config
        orawsp:Silent = "true"
        orawsp:Enforced = "true"
        orawsp:name = "STS Trust Configuration"
        orawsp:category = "security/sts-config"
        orasp:wsdl-uri = "http://host:port/sts?wsdl"
        orasp:port-uri = "http://host:port/sts-service"
orasp:port-endpoint="target-namespace#wsdl.endpoint(service-name/port-name)"
        orasp:policy-reference-uri="oracle/policy-name"
        orasp:soap-version="12"
        orasp:sts-keystore-recipient-alias="sts-csf-key">
        <orawsp:bindings>
            <orawsp:Config orawsp:name = "StsTrustConfig" orawsp:configType =
"declarative">
             <orawsp:PropertySet orawsp:name="standard-security-properties">
               <orawsp:Property orawsp:name="role" orawsp:type="string"
orawsp:contentType="constant">
                  <orawsp:Value>ultimateReceiver</orawsp:Value>
               </orawsp:Property>
             </orawsp:PropertySet>
            </orawsp:Config>
        </orawsp:bindings>
    </orasp:sts-trust-config>
</wsp:Policy>
```

11.8.2.2 Settings

You can change the settings shown in Table C–96.

11.8.2.3 Configuration Properties

You can configure the properties shown in Table C–97.

11.8.2.4 How to Set Up the Web Service Client

You are encouraged to configure the STS config policy from the Web service, as described in "Setting Up Automatic Policy Configuration for STS" on page 10-104.

However, if you did not configure the STS config policy from the Web service, or if you are using the SAML sender vouches confirmation method, you must then configure it from the Web service client.   See "Manually Configuring the STS Config Policy From the Web Service Client: Main Steps" on page 10-107 for information on how to set up the Web service client.

11.8.2.5 How to Set Up the Web Service Client at Design Time

You are encouraged to configure the STS config policy from the Web service, as described in "Setting Up Automatic Policy Configuration for STS" on page 10-104.

However, if you did not configure the STS config policy from the Web service, or if you are using the SAML sender vouches confirmation method, you must then configure it from the Web service client as described in this section.

See "Manually Configuring the STS Config Policy From the Web Service Client: Main Steps" on page 10-107 for additional information on how to set up the Web service client.

You can set up and attach the `oracle/sts_trust_config_client_policy` policy programmatically, as shown in Example 11–6 and Example 11–7.

Example 11–6   Sample JSE Proxy Client

```
URL endpointUrl = new URL(getWebConnectionString() +
"/jaxws-test-service/jaxws-test-port");

ServiceDelegateImpl client = new ServiceDelegateImpl(
    new  URL(endpointUrl.toString() + "?WSDL"),
    new QName("http://jaxws.oracle.com/targetNamespace/JaxwsService",
"JaxwsService"),
    OracleService.class);

JaxwsService port = client.getPort(
    new  QName("http://jaxws.oracle.com/targetNamespace/JaxwsService",
"JaxwsServicePort"),
    test.jaxws.client.JaxwsService.class);

((BindingProvider)port).getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_
PROPERTY,endpointUrl.toExternalForm());
((BindingProvider)port).getRequestContext().put(ClientConstants.CLIENT_CONFIG,
    fileToElement(new File("./jaxws/client/dat/oracle-webservice-client.xml")));
```

The related `oracle-webservice-client.xml` file with the STS config policy and STS issue policy is shown in Example 11–7.

Example 11–7 Sample oracle-webservice-client.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<oracle-webservice-clients>
    <webservice-client>
        <port-info>
            <policy-references>
                <policy-reference uri="oracle/sts_trust_config_client_policy"
category="security"/>
                <policy-reference uri="oracle/wss11_sts_issue_saml_hok_with_
message_protection_client_policy " category="security"/>
             </policy-references>
        </port-info>
    </webservice-client>
</oracle-webservice-clients>
```

11.8.2.6 How to Set Up the Web Service

See "Setting Up Automatic Policy Configuration for STS" on page 10-104 for the steps to follow.

11.8.3 oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy

This policy inserts a SAML bearer assertion issued by a trusted STS. Messages are protected using SSL.

This policy contains the following assertion template: oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy_template. See "WS-Trust Assertion Templates" on page C-164 for more information about the assertion.

11.8.3.1 Policy Assertion

The `oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy` assertion is as follows:

```
<orasp:wss-sts-issued-token-over-ssl
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orasp:require-applies-to="true" orasp:require-client-entropy="true"
 orasp:require-server-entropy="true" orasp:trust-version="13"
 orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WS-Security 1.1, issued token over ssl">
<orasp:issued-token orasp:require-external-reference="true"
 orasp:require-internal-reference="true" orasp:use-derived-keys="false">
<orasp:request-security-token-template orasp:key-type="Bearer"
orasp:token-type="SAML11"/>
</orasp:issued-token>
<orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
```

```
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
orawsp:name="WssStsIssuedTokenOverSSLConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
<orawsp:Property orawsp:contentType="optional" orawsp:name="sts.auth.user.csf.key"
orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional" orawsp:name="sts.auth.x509.csf.key"
orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:name="on.behalf.of" orawsp:type="boolean">
<orawsp:Value>false</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.on.behalf.of.csf.key" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.service.principal.name" orawsp:type="string">
<orawsp:Value>HOST/localhost@EXAMPLE.COM</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.keytab.location" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.caller.principal.name" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:wss-sts-issued-token-over-ssl>
```

11.8.3.2 Settings

You can change the settings shown in Table C–100.

11.8.3.3 Configuration Properties

You can configure the properties shown in Table C–101.

11.8.3.4 How to Set Up the Web Service Client

See "Setting Up Automatic Policy Configuration: Main Steps" on page 10-105 for information on how to set up the Web service client.

This policy requires you to set up the Oracle WSM keystore to specify a key (username/password or X.509) to authenticate to the STS. See "Configuring Keystores for Message Protection" on page 10-9.

See "Programmatic Configuration Overrides for WS-Trust Client Policies" on page 10-110 for a description of the configuration settings you can override.

If you do not set **Require Mutual Authentication**, one-way SSL is involved. See "Configuring SSL for a Web Service Client" on page 10-41.

If you do set **Require Mutual Authentication**, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

11.8.3.5 How to Set Up the Web Service Client at Design Time

See "Setting Up Automatic Policy Configuration: Main Steps" on page 10-105 for information on how to set up the Web service client.

This policy requires you to set up the Oracle WSM keystore to specify a key (username/password or X.509) to authenticate to the STS. See "Configuring Keystores for Message Protection" on page 10-9.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override. See "Programmatic Configuration Overrides for WS-Trust Client Policies" on page 10-110 for examples of overriding STS configuration settings.

If you do not set **Require Mutual Authentication**, one-way SSL is involved. See "Configuring SSL for a Web Service Client" on page 10-41.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 10-42.

11.8.4 oracle/wss_saml_bearer_or_username_token_service_policy

This policy enforces one of the following authentication policies, based on whether the client uses a SAML or username token, respectively:

- SAML token within WS-Security SOAP header using the bearer confirmation type.
- WS-Security UsernameToken SOAP header to authenticate users against the configured identity store.

This policy contains the following assertions as an OR group—meaning either type of policy can be enforced by a client:

- oracle/wss_saml_token_bearer_template. See "oracle/wss_saml_token_bearer_service_template" on page C-73 for more information about the assertion.
- oracle/wss_username_token_template. See ""oracle/wss_username_token_service_template" on page C-33 for more information about the assertion.

11.8.5 oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy

This policy authenticates a SAML bearer assertion issued by a trusted STS. Messages are protected using SSL.

This policy contains the following assertion template: oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy_template. See "WS-Trust Assertion Templates" on page C-164 for more information about the assertion.

11.8.5.1 Policy Assertion

The `oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy` assertion is as follows:

```
<orasp:wss-sts-issued-token-over-ssl
xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
orasp:require-applies-to="true" orasp:require-client-entropy="true"
```

```
orasp:require-server-entropy="true" orasp:trust-version="13"
orawsp:Enforced="true" orawsp:Silent="false"
orawsp:category="security/authentication, security/msg-protection"
orawsp:name="WS-Security 1.1, issued token over ssl">
<orasp:issued-token orasp:require-external-reference="true"
orasp:require-internal-reference="true" orasp:use-derived-keys="false">
<orasp:request-security-token-template orasp:key-type="Bearer"
orasp:token-type="SAML11"/>
</orasp:issued-token>
<orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
orawsp:name="WssStsIssuedTokenOverSSLConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
<orawsp:Property orawsp:contentType="constant" orawsp:name="role"
orawsp:type="string">
<orawsp:Value>ultimateReceiver</orawsp:Value>
</orawsp:Property>
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:wss-sts-issued-token-over-ssl>
```

11.8.5.2 Settings

See Table C–100.

11.8.5.3 Configuration Properties

You can configure the properties shown in Table C–102.

11.8.5.4 How to Set Up the Web Service

See "Setting Up Automatic Policy Configuration: Main Steps" on page 10-105 for information on how to set up the Web service.

To configure SSL, see "Configuring SSL on WebLogic Server (One-Way)" on page 10-39, or "Configuring SSL on WebLogic Server (Two-Way)" on page 10-40 if **Require Mutual Authentication** is checked.

11.8.6 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy

This policy inserts a SAML HOK assertion issued by a trusted STS (Security Token Service). Messages are protected using proof key material provided by the STS.

This policy contains the following assertion template: oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template. See "WS-Trust Assertion Templates" on page C-164 for more information about the assertion.

11.8.6.1 Policy Assertion

The oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy assertion is as follows:

```
<orasp:wss11-sts-issued-token-with-certificates
xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
orasp:require-applies-to="true" orasp:require-client-entropy="true"
orasp:require-server-entropy="true" orasp:trust-version="13"
orawsp:Enforced="true" orawsp:Silent="false"
orawsp:category="security/authentication, security/msg-protection"
```

```
orawsp:name="WS-Security 1.1, issued tokee">
<orasp:issued-token orasp:require-external-reference="true"
orasp:require-internal-reference="true" orasp:use-derived-keys="false">
<orasp:request-security-token-template orasp:algorithm-suite="Basic128"
orasp:key-type="Symmetric" orasp:token-type="SAML11"/>
</orasp:issued-token>
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint" orasp:is-encrypted="false"
orasp:is-signed="true" orasp:sign-key-ref-mech="thumbprint"/>
<orasp:msg-security orasp:algorithm-suite="Basic128"
orasp:confirm-signature="true" orasp:encrypt-signature="false"
orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
orasp:use-derived-keys="false">
<orasp:request>
<orasp:signed-parts>
<orasp:body/>
<orasp:header orasp:namespace="http://www.w3.org/2005/08/addressing"/>
<orasp:header orasp:namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
<orasp:header orasp:name="fmw-context"
orasp:namespace="http://xmlns.oracle.com/fmw/context/1.0"/>
</orasp:signed-parts>
<orasp:encrypted-parts>
<orasp:body/>
<orasp:header orasp:name="fmw-context"
orasp:namespace="http://xmlns.oracle.com/fmw/context/1.0"/>
</orasp:encrypted-parts>
</orasp:request>
<orasp:response>
<orasp:signed-parts>
<orasp:body/>
</orasp:signed-parts>
<orasp:encrypted-parts>
<orasp:body/>
</orasp:encrypted-parts>
</orasp:response>
<orasp:fault/>
</orasp:msg-security>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
orawsp:name="Wss11StsIssuedTokenWithCertsConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
<orawsp:Property orawsp:contentType="optional" orawsp:name="sts.auth.user.csf.key"
orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional" orawsp:name="sts.auth.x509.csf.key"
orawsp:type="string">
<orawsp:Value>enc-csf-key</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:name="on.behalf.of" orawsp:type="boolean">
<orawsp:Value>false</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.on.behalf.of.csf.key" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:name="keystore.recipient.alias" orawsp:type="string">
<orawsp:Value>orakey</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional" orawsp:name="keystore.enc.csf.key"
orawsp:type="string">
```

```
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.service.principal.name" orawsp:type="string">
<orawsp:Value>HOST/localhost@EXAMPLE.COM</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.keytab.location" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.caller.principal.name" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:wss11-sts-issued-token-with-certificates>
```

11.8.6.2 Settings

You can change the settings shown in Table C–103.

11.8.6.3 Configuration Properties

You can configure the properties shown in Table C–104.

11.8.6.4 How to Set Up the Web Service Client

See "Setting Up Automatic Policy Configuration: Main Steps" on page 10-105 for information on how to set up the Web service client.

This policy requires you to set up the Oracle WSM keystore to specify a key (username/password or X.509) to authenticate to the STS. See "Configuring Keystores for Message Protection" on page 10-9.

See "Programmatic Configuration Overrides for WS-Trust Client Policies" on page 10-110 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias`, or override it on a per-client basis when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.enc.csf.key`, or override them on a per-client basis when you attach the policy.

11.8.6.5 How to Set Up the Web Service Client at Design Time

See "Setting Up Automatic Policy Configuration: Main Steps" on page 10-105 for information on how to set up the Web service client.

This policy requires you to set up the Oracle WSM keystore to specify a key (username/password or X.509) to authenticate to the STS. See "Configuring Keystores for Message Protection" on page 10-9.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override. See "Programmatic Configuration Overrides for WS-Trust Client Policies" on page 10-110 for examples of overriding STS configuration settings.

Configure the policy assertion for message signing, message encryption, or both.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias`, or override it on a per-client basis when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.enc.csf.key`, or override them on a per-client basis when you attach the policy.

11.8.7 oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy

This policy authenticates a SAML HOK assertion issued by a trusted STS (Security Token Service). Messages are protected using WS-Security's Basic 128 suite of symmetric key technologies.

This policy contains the following assertion template: oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template. See "WS-Trust Assertion Templates" on page C-164 for more information about the assertion.

11.8.7.1 Policy Assertion

The oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy assertion is as follows:

```
<orasp:wss11-sts-issued-token-with-certificates
xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
orasp:require-applies-to="true" orasp:require-client-entropy="true"
orasp:require-server-entropy="true" orasp:trust-version="13"
orawsp:Enforced="true" orawsp:Silent="false"
orawsp:category="security/authentication, security/msg-protection"
orawsp:name="WS-Security 1.1, issued tokee">
<orasp:issued-token orasp:require-external-reference="true"
orasp:require-internal-reference="true" orasp:use-derived-keys="false">
<orasp:request-security-token-template orasp:algorithm-suite="Basic128"
orasp:key-type="Symmetric" orasp:token-type="SAML11"/>
</orasp:issued-token>
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint" orasp:is-encrypted="false"
orasp:is-signed="true" orasp:sign-key-ref-mech="thumbprint"/>
<orasp:msg-security orasp:algorithm-suite="Basic128"
orasp:confirm-signature="true" orasp:encrypt-signature="false"
orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
orasp:use-derived-keys="false">
<orasp:request>
<orasp:signed-parts>
<orasp:body/>
<orasp:header orasp:namespace="http://www.w3.org/2005/08/addressing"/>
<orasp:header orasp:namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
<orasp:header orasp:name="fmw-context"
orasp:namespace="http://xmlns.oracle.com/fmw/context/1.0"/>
</orasp:signed-parts>
```

```
<orasp:encrypted-parts>
<orasp:body/>
<orasp:header orasp:name="fmw-context"
orasp:namespace="http://xmlns.oracle.com/fmw/context/1.0"/>
</orasp:encrypted-parts>
</orasp:request>
<orasp:response>
<orasp:signed-parts>
<orasp:body/>
</orasp:signed-parts>
<orasp:encrypted-parts>
<orasp:body/>
</orasp:encrypted-parts>
</orasp:response>
<orasp:fault/>
</orasp:msg-security>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
orawsp:name="Wss11StsIssuedTokenWithCertsConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
<orawsp:Property orawsp:contentType="optional" orawsp:name="keystore.enc.csf.key"
orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="constant" orawsp:name="role"
orawsp:type="string">
<orawsp:Value>ultimateReceiver</orawsp:Value>
</orawsp:Property>
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:wss11-sts-issued-token-with-certificates>
```

11.8.7.2 Settings

You can change the settings shown in Table C–103.

11.8.7.3 Configuration Properties

You can configure the properties shown in Table C–105. You also have the option to override the keystore.enc.csf.key server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.

11.8.7.4 How to Set Up the Web Service

See "Setting Up Automatic Policy Configuration: Main Steps" on page 10-105 for information on how to set up the Web service.

11.8.8 oracle/wss11_sts_issued_saml_with_message_protection_client_policy

This policy inserts a SAML sender vouches assertion issued by a trusted STS (Security Token Service). Messages are protected using the client's private key.

This policy contains the following assertion template: oracle/wss11_sts_issued_saml_with_message_protection_client_policy. See "WS-Trust Assertion Templates" on page C-164 for more information about the assertion.

11.8.8.1 Policy Assertion

The oracle/wss11_sts_issued_saml_with_message_protection_client_policy policy assertion is as follows:

```
<orasp:wss11-sts-issued-token-with-certificates
xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
orasp:require-applies-to="true" orasp:require-client-entropy="true"
orasp:require-server-entropy="true" orasp:trust-version="13"
orawsp:Enforced="true" orawsp:Silent="false"
orawsp:category="security/authentication, security/msg-protection"
orawsp:name="WS-Security 1.1, issued token">
<orasp:issued-token orasp:require-external-reference="true"
orasp:require-internal-reference="true" orasp:use-derived-keys="false">
<orasp:request-security-token-template orasp:algorithm-suite="Basic128"
orasp:token-type="SAML11"/>
</orasp:issued-token>
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint" orasp:is-encrypted="false"
orasp:is-signed="true" orasp:sign-key-ref-mech="direct"/>
<orasp:msg-security orasp:algorithm-suite="Basic128"
orasp:confirm-signature="true" orasp:encrypt-signature="false"
orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
orasp:use-derived-keys="false">
<orasp:request>
<orasp:signed-parts>
<orasp:body/>
<orasp:header orasp:namespace="http://www.w3.org/2005/08/addressing"/>
<orasp:header orasp:namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
<orasp:header orasp:name="fmw-context"
orasp:namespace="http://xmlns.oracle.com/fmw/context/1.0"/>
</orasp:signed-parts>
<orasp:encrypted-parts>
<orasp:body/>
<orasp:header orasp:name="fmw-context"
orasp:namespace="http://xmlns.oracle.com/fmw/context/1.0"/>
</orasp:encrypted-parts>
</orasp:request>
<orasp:response>
<orasp:signed-parts>
<orasp:body/>
</orasp:signed-parts>
<orasp:encrypted-parts>
<orasp:body/>
</orasp:encrypted-parts>
</orasp:response>
<orasp:fault/>
</orasp:msg-security>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
orawsp:name="Wss11StsIssuedTokenWithCertsConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
<orawsp:Property orawsp:contentType="optional" orawsp:name="sts.auth.user.csf.key"
orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional" orawsp:name="sts.auth.x509.csf.key"
orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:name="on.behalf.of" orawsp:type="boolean">
```

```
<orawsp:Value>true</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.on.behalf.of.csf.key" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="keystore.recipient.alias" orawsp:type="string">
<orawsp:Value>orakey</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional" orawsp:name="keystore.enc.csf.key"
orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:wss11-sts-issued-token-with-certificates>
```

11.8.8.2 Settings

You can change the settings shown in Table C–106.

11.8.8.3 Configuration Properties

You can configure the properties shown in Table C–107.

11.8.8.4 How to Set Up the Web Service Client

See "Setting Up Automatic Policy Configuration: Main Steps" on page 10-105 for information on how to set up the Web service client.

This policy requires you to set up the Oracle WSM keystore to specify a key (username/password or X.509) to authenticate to the STS. See "Configuring Keystores for Message Protection" on page 10-9.

See "Programmatic Configuration Overrides for WS-Trust Client Policies" on page 10-110 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias`, or override it on a per-client basis when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.enc.csf.key`, or override them on a per-client basis when you attach the policy.

11.8.8.5 How to Set Up the Web Service Client at Design Time

See "Setting Up Automatic Policy Configuration: Main Steps" on page 10-105 for information on how to set up the Web service client.

This policy requires you to set up the Oracle WSM keystore to specify a key (username/password or X.509) to authenticate to the STS. See "Configuring Keystores for Message Protection" on page 10-9.

See "Using Client Programmatic Configuration Overrides" on page 11-128 for a description of the configuration settings you can override. See "Programmatic Configuration Overrides for WS-Trust Client Policies" on page 10-110 for examples of overriding STS configuration settings.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 10-57.

As an alternative, you can specify a value for `keystore.recipient.alias`, or override it on a per-client basis when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for `keystore.enc.csf.key`, or override them on a per-client basis when you attach the policy.

Configure the policy assertion for message signing, message encryption, or both.

11.9 MTOM Attachment Policies and Configuration Steps

This section describes the predefined MTOM policies.

There are two potential behaviors for using MTOM with Oracle Infrastructure Web services, depending on how you configure MTOM:

- If you configure MTOM from Fusion Middleware Control by attaching the oracle/wsmtom_policy policy (either via local or Global Policy Attachment), the endpoint throws a fault if the request is not MTOM encoded. The MTOM policy rejects inbound messages that are not in MTOM format and verifies that outbound messages are in MTOM format. In this use, requests must be MTOM-enabled.
- If you configure MTOM for an ADF BC Web service outside of Fusion Middleware Control, such as by editing the MTOM-enabled switch in `oracle-webservices.xml` or by directly adding the @MTOM annotation to the Web service, the endpoint can accept MTOM requests but does not return a fault if the request is not MTOM encoded. In this use, requests might be MTOM-enabled, but there is no requirement that they must be.

11.9.1 oracle/wsmtom_policy

SOAP Message Transmission Optimization Mechanism/XML-binary Optimized Packaging (MTOM/XOP) defines a method for optimizing the transmission of XML data of type xs:base64Binary or xs:hexBinary in SOAP messages.

The Message Transmission Optimization Mechanism (MTOM) policy rejects inbound messages that are not in MTOM format and verifies that outbound messages are in MTOM format.

MTOM refers to specifications `http://www.w3.org/TR/2005/REC-soap12-mtom-20050125` and `http://www.w3.org/Submission/2006/SUBM-soap11mtom10-20060405` for SOAP 1.2 and SOAP 1.1 bindings, respectively.

11.9.1.1 How to Set Up the Web Service Client

No configuration is required.

11.9.1.2 How to Set Up the Web Service Client at Design Time

To enable MTOM on the client of the Web service, pass the javax.xml.ws.soap.MTOMFeature as a parameter when creating the Web service proxy or dispatch, as illustrated in the following example.

```
package examples.webservices.mtom.client;
import javax.xml.ws.soap.MTOMFeature;
public class Main {
  public static void main(String[] args) {
    String FOO = "FOO";
    MtomService service = new MtomService()
    MtomPortType port = service.getMtomPortTypePort(new MTOMFeature());
    String result = null;
    result = port.echoBinaryAsString(FOO.getBytes());
    System.out.println( "Got result: " + result );
  }
}
```

11.9.1.3 How to Set Up Oracle Platform Security Services (OPSS)

No configuration is required.

11.10 Reliable Messaging Policies and Configuration Steps

WS-ReliableMessaging makes message exchanges reliable. It ensures that messages are delivered reliably between distributed applications regardless of software component, system, or network failures. Ordered delivery is assured and automatic retransmission of failed messages does not have to be coded by each client application.

Consider using reliable messaging if your Web service is experiencing the following problems:

- network failures or dropped connections
- messages are lost in transit
- messages are arriving at their destination out of order

WS-ReliableMessaging considers the source and destination of a message to be independent of the client/server model. That is, the client and the server can each act simultaneously as both a message source and destination on the communications path.

This section describes the predefined Reliable Messaging policies.

11.10.1 WS-RM Policy Properties

Table 11–1 lists the properties that you can set for the WS-RM policies.

Table 11–1 WS-RM Policy Properties

Property Name	Description	Default Value Used by Policy	Possible Values
DeliveryAssurance	Delivery assurance. The following defines the delivery assurance types: ■ At Most Once—Messages are delivered at most once, without duplication. ■ At Least Once—Every message is delivered at least once. It is possible that some messages are delivered more than once. ■ Exactly Once—Every message is delivered exactly once, without duplication. ■ Messages are delivered in the order that they were sent. This delivery assurance can be combined with one of the preceding three assurances.	InOrder	InOrder AtLeastOnce AtLeastOnceInOrder ExactlyOnce ExactlyOnceInOrder AtMostOnce AtMostOnceInOrder
StoreType	Type of message store.	InMemory	InMemory FileSystem (not fully supported) JDBC
StoreName	Name of the message store.	oracle	String value
jdbc-connection-name	JNDI reference to a JDBC data source. This field is valid only if StoreType is set to JDBC. This value takes precedence over jdbc-connection-url. The username and password will be used if both are present.	jdbc/MessagesStore	Valid JDBC store
InactivityTimeout	Amount of time, in milliseconds, that can elapse between message exchanges associated with a particular WS-ReliableMessaging sequence. Once this value is reached, the sequence will be terminated and discarded automatically.	600000	The amount of time in milliseconds.
BaseRetransmissionInterval	Interval, in milliseconds, that the source endpoint waits after transmitting a message and before it retransmits the message if it receives no acknowledgment for that message.	3000	The amount of time in milliseconds.

11.10.2 oracle/wsrm10_policy

This policy provides support for version 1.0 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint.

11.10.2.1 How to Set Up the Web Service Client

The Web service client will automatically detect the WSDL policy assertions at run time and use them to enable the advertised version of WS-RM on the client.

11.10.2.2 How to Set Up the Web Service Client at Design Time

For multi-message sequences, the client code must include explicit invocations of methods for delimiting sequence boundaries. Otherwise, every message is wrapped in its own sequence

Edit the client to enable a reliable messaging session for the messages sent to the service. The *oracle.webservices.rm.client.RMSessionLifecycle* interface provides the client with a mechanism for demarcating WS-RM sequence boundaries.

Example 11–8 illustrates sample WS-RM client code. In the code, a new TestService is created. The TestPort, through which the client will communicate with the service, is retrieved. The port object is cast to a *RMSessionLifecycle* object and a reliable messaging session is opened on it (*openSession*). After the messages are sent to the service, the session is closed (*closeSession*).

Example 11–8 Sample WS-Rm Client Code

```
public class ClientServlet extends HttpServlet {

    public void doGet(HttpServletRequest request,
                HttpServletResponse response) throws ServletException,
                                                     IOException {

        int num1 =  Integer.parseInt(request.getParameter("num1"));
        int num2 =  Integer.parseInt(request.getParameter("num2"));
        String outputStr = null;

        TestService service = new TestService();
        Test port = service.getTestPort();

        try {
        ((RMSessionLifecycle) port).openSession();
            outputStr = port.hello(inputStr);
        } catch (Exception e) {
            e.printStackTrace();
            outputStr = e.getMessage();
        } finally {
        ((RMSessionLifecycle) port).closeSession();
            response.getOutputStream().write(outputStr.getBytes());
        }
    }
}
```

11.10.2.3 How to Set Up Oracle Platform Security Services (OPSS)

No additional configuration is required.

11.10.3 oracle/wsrm11_policy

This policy provides support for version 1.1 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint.

11.10.3.1 How to Set Up the Web Service Client

The Web service client will automatically detect the WSDL policy assertions at run time and use them to enable the advertised version of WS-RM on the client.

11.10.3.2 How to Set Up the Web Service Client at Design Time

For multi-message sequences, the client code must include explicit invocations of methods for delimiting sequence boundaries. Otherwise, every message is wrapped in its own sequence

Edit the client to enable a reliable messaging session for the messages sent to the service. The *oracle.webservices.rm.client.RMSessionLifecycle* interface provides the client with a mechanism for demarcating WS-RM sequence boundaries.

Example 11–8 illustrates a servlet client. In the code, a new TestService is created. The TestPort, through which the client will communicate with the service, is retrieved. The port object is cast to a *RMSessionLifecycle* object and a reliable messaging session is opened on it (*openSession*). After the messages are sent to the service, the session is closed (*closeSession*).

11.10.3.3 How to Set Up Oracle Platform Security Services (OPSS)

No additional configuration is required.

11.11 Management Policies and Configuration Steps

This section describes the predefined Management policies.

11.11.1 oracle/log_policy

This policy causes the request, response, and fault messages to be sent to a message log.

This policy contains the following assertion template: *oracle/log_template*. See "oracle/security_log_template" on page C-195 for more information about the assertion.

11.11.1.1 Settings

See Table C–120.

11.11.1.2 Configuration Properties

None defined.

11.11.1.3 How to Set Up the Web Service or Client

Determine whether you want to log messages for the request and response, based on the following categories:

- all
- header
- SOAP body
- SOAP envelope

11.11.1.4 How to Set Up Oracle Platform Security Services (OPSS)

Messages are logged to the message log for the domain.

To view the message log

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you want to see the logged messages.   Select the domain.

2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Logs** and then **View Log Messages**.

11.12 Attaching Policy Files to Web Services and Clients

There are two ways to attach policies to Web service clients and Web services: at the client and service design time, and post deployment.

Post-deployment, you attach security and management policies to SOA composites, ADF, and WebCenter applications using the Oracle Enterprise Manager Fusion Middleware Control. This method provides the most power and flexibility because it moves Web service security to the control of the security administrator.

At design time, Oracle JDeveloper automates ADF and SOA client policy attachment. Or, you can attach Oracle WSM security and management policies to applications programmatically. You typically do this using your favorite IDE, such as Oracle JDeveloper.

Either way, the client-side policy must be the equivalent of the one associated with the Web service. If the two files are different, and there is a conflict in the assertions contained in the files, then the invoke of the Web service operation returns an error.

For example, if the oracle/wss_http_token_over_ssl_service_policy policy requires mutual authentication, the client policy must also be set for mutual authentication.

For the predefined policies, both client and Web service policies are included. If you create a new policy, generating the policy as described in "Creating Web Service Policies" on page 7-4 increases the likelihood that the client policy will work with the service policy.

11.13 Using Client Programmatic Configuration Overrides

"Attaching Client Policies Permitting Overrides" on page 8-31 describes the policy configuration override feature that allows you to specify certain Web service client configuration information when you attach a policy. However, you can also override this configuration information programmatically at design time. This section describes client programmatic overrides.

Table 11–2 shows the properties you can set via programmatic configuration overrides for a given policy. Example 11–9 shows an example of setting these properties from a program.

Table 11–2 Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.CALLER_PRINCIPAL_NAME	Client's principal name as generated using the `ktpass` command and mapped to the username for which the kerberos token should be generated. Use the following format: `<username>@<REALM NAME>`. **Note:** `keytab.location` and `caller.principal.name` are required for propagating client identity for Java EE applications.	wss11_kerberos_token_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_CSF_KEY	Gets the username and password corresponding to the csf-key specified in the credential store if the credential store is available to the client. Instead of using this property, you can also explicitly set the username and password as shown in Example 11–9	oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss_username_token_client_policy oracle/wss_username_token_over_ssl_client_policy oracle/wss10_username_id_propagation_with_msg_protection_client_policy oracle/wss_http_token_client_policy oracle/wss_http_token_over_ssl_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_LOCATION	This property sets the location of the keystore file. If provided, this value will override any statically configured value. Type: java.lang.String	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_TYPE	This property sets the type of keystore file. If provided, this value will override any statically configured value. Type: java.lang.String Default is JKS. Can also be KSS.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_PASSWORD	This property sets the password of the keystore file. If provided, this value will override any statically configured value. Type: java.lang.String	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SIG_KEY_ALIAS	This property sets the alias of the key within the keystore that will be used for digital signatures. If provided, this value will override any statically configured value. Type: java.lang.String For WSS11 policies, this property is used only in the case of mutual authentication.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SIG_KEY_PASSWORD	This property sets the password for the alias of the key within the keystore that will be used for digital signatures. If provided, this value will override any statically configured value. Type: java.lang.String For WSS11 policies, this property is used only in the case of mutual authentication.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_ENC_KEY_ALIAS	This property sets the alias of the key within the keystore that will be used to decrypt the response from the service. If provided, this value will override any statically configured value. Type: java.lang.String Not used in WSS11 policies.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_ENC_KEY_PASSWORD	This property sets the password for the key within the keystore that will be used for decryption. If provided, this value will override any statically configured value. Type: java.lang.String Not used in WSS11 policies.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_RECIPIENT_KEY_ALIAS	This property sets the alias for the recipient's public key that is used to encrypt type outbound message. If provided this value will override any static configuration value. Type: java.lang.String	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SUBJECT_PRECEDENCE	In case of SAML client policies, set this property to false if there is a need to use a client-specified username rather than subject.	Applies to all of the SAML client policies listed in "Configuring SAML" on page 10-64.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SAML_ISSUER_NAME	This property sets the SAML issuer name when trying access a service that is protected using SAML mechanism. If provided this value will override any static configuration value. Type: java.lang.String	Applies to all of the SAML client policies listed in "Configuring SAML" on page 10-64.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_INCLUDE_USER_ROLES	This property sets the user roles in a SAML assertion.	Applies to all of the SAML client policies listed in "Configuring SAML" on page 10-64.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SAML_ASSERTION_FILE_NAME	For SAML HOK policies, this file contains the assertion	Applies to all of the SAML client policies listed in "Configuring SAML" on page 10-64.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KERBEROS_SERVICE_PRINCIPAL	This property sets the service principal name when trying access a service that is protected using the Kerberos mechanism. If provided this value will override any static configuration value. Type: java.lang.String	oracle/wss11_kerberos_token_with_message_protection_client_policy
BindingProvider.USERNAME_PROPERTY (`javax.xml.ws.security.auth.username`)	User name for authentication.	Used by username policies, and SAML policies including identity switching policies. For username client policies, you have two options: ▪ csf-key ▪ `BindingProvider.USERNAME_PROPERTY` and `BindingProvider.PASSWORDproperty`. For SAML client policies including the identity switch policy, use `BindingProvider.USERNAME_PROPERTY`.
BindingProvider.PASSWORD_PROPERTY (`javax.xml.ws.security.auth.password`)	Password for authentication.	Used by username client policies.

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY	Use to configure X509 certificate for authenticating to the STS. If the `policy-reference-uri` in the STS configuration policy points to an x509-based policy, then you configure the `sts.auth.x509.csf.key` property to specify the X509 certificate for authenticating to the STS.	oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy oracle/wss11_sts_issued_saml_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_STS_AUTH_USER_CSF_KEY	Use to configure the username/password to authenticate to the STS. If `policy-reference-uri` in the STS configuration policy points to a username-based policy, then you configure the `sts.auth.user.csf.key` property to specify a username/password to authenticate to the STS.	oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy oracle/wss11_sts_issued_saml_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_STS_AUTH_ON_BEHALF_OF_CSF_KEY	Optional property. Use to configure on behalf of entity. If present, it will be given preference over Subject (if it exists).	oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy oracle/wss11_sts_issued_saml_with_message_protection_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.ON_BEHALF_OF	Optional property. Override this property to indicate whether the request is on behalf of an another entity. The default value for this flag is true. When set to true and `sts.auth.on.behalf.of.csf.key` is configured, then it will be given preference and the identity established using that CSF key will be send in the on behalf of. Otherwise, if the subject is already established, then the username from the subject will be sent as `onBehalfOf` token. If `sts.auth.on.behalf.of.csf.key` is not set and the subject does not exist, `on.behalf.of` is treated as a token exchange for the requestor and not for another entity. It is not included in an `onBehalfOf` element in the request.	oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy oracle/wss11_sts_issued_saml_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.STS_KEYSTORE_RECIPIENT_ALIAS	The public key alias of the STS.	oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy oracle/wss11_sts_issued_saml_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.ATTESTING_MAPPING_ATTRIBUTE	The mapping attribute used to represent the attesting entity. Only the DN is currently supported. This attribute is applicable only to sender vouches and then only to message protection use cases. It is not applicable to SAML over SSL policies.	wss10_saml20_token_with_message_protection_client_policy wss11_saml20_token_with_message_protection_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.SAML_AUDIENCE_URI	Represents the relying party, as a comma-separated URI. This field accepts wildcards.	wss10_saml_token_client_policy wss10_saml20_token_client_policy wss_saml_token_bearer_over_ssl_client_policy wss_saml20_token_bearer_over_ssl_client_policy wss_saml_token_over_ssl_client_policy wss_saml20_token_over_ssl_client_policy wss10_saml_token_with_message_protection_client_policy wss10_saml20_token_with_message_protection_client_policy wss11_saml_token_with_message_protection_client_policy wss11_saml20_token_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_OAUTH2_CLIENT_CSF_KEY	Required property that specifies the key to use to obtain the client username and password.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_AUDIENCE_URI	Audience restriction. The following conditions are supported: ■ If this property is not set, the service URL is used as the audience URI ■ If this property is set to NONE (not case sensitive), then the audience URI is set to null. ■ If this property is set to a value other than NONE, then the audience URI is set to this value.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_AUTHORIZATION_CODE	Optional property for passing the authorization code for the 3-legged OAuth2 use case. (Not supported in this release.)	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_FEDERATED_CLIENT_TOKEN	Optional property which, by default, specifies that a JWT token is generated for the client using the values of the oauth2.client.csf.key and keystore.sig.csf.key properties. If set to false, the oauth2.client.csf.key is used to generate an Authorization header to be sent in the client request to the OAUTH2 server.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_INCLUDE_CERTIFICATE	When true, the signature certificate and the trusted certificate chain (for CA-issued certificates) are included in JWT token claim. This increases the size of the JWT token, but you do not need to then import the certificate and certificate chain into the service side keystore. When false, only the thumbprint and alias of the certificate are included in the JWT token.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_ISSUER_NAME	Optional property that specifies the issuer name used for the locally-generated JWT token (iss:claim). By default it is www.oracle.com.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_INCLUDE_USER_ROLES	Optional property that specifies whether the user roles from the Subject are included in the JWT token as claims.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_PROPAGATE_IDENTITY_CONTEXT	Optional property that specifies whether the identity context information is propagated as claims in the JWT token.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_REDIRECT_URI	Optional property that specifies the redirect URIs that the OAuth server will use to redirect the user-agent to the client once access is granted or denied.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy

Table 11–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SCOPE	Optional property that specifies the scope (as-is) of the OAuth2 request. If present, the scope is included in the OAuth2 token request with this value.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_TOKEN_URI	Required property that specifies the token endpoint of the OAuth2 server.	oauth2_config_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_USER_ATTRIBUTES	Optional property that specifies whether user attributes are inserted as claims in JWT token.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_USER_TENANT_NAME	Reserved for use with Oracle Cloud.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_CSF_MAP	Oracle WSM map in the credential store that contains the CSF aliases.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SIG_CSF_KEY	Optional property that specifies the tenant key from the Oracle WSM keystore for signing the locally-created JWT token.	http_oauth2_token_client_policy http_oauth2_token_over_ssl_client_policy

11.13.1 Configuration Override Example

Example 11–9 shows an example of a Web service client overriding the keystore and username/password.

If you need to clear an overridden configuration property, set it to an empty string.

Before you clear it, remember that other policies could be using the same property. The properties are client-specific and there could be multiple policies that are attached to the same client that use the same property.

Example 11–9 Overriding the Keystore and Username/Password

```
package example;
import oracle.wsm.security.utils.SecurityConstants;
public class MyClientJaxWs {
    public static void main(String[] args) {
        try {
            URL serviceWsdl = new URL("http://localhost/myApp/myPort?WSDL");
            QName serviceName = new QName("MyNamespace", "MyService");
            Service service = Service.create(serviceWsdl, serviceName);
            MyInterface proxy = service.getPort(MyInterface.class);
            RequestContext context = ((BindingProvider)proxy).getRequestContext();
            context.put(oracle.webservices.ClientConstants.CLIENT_CONFIG, new
File( "c:/dat/client-pdd.xml" ) );
            context.put(BindingProvider.USERNAME_PROPERTY, getCurrentUsername() );
            context.put(BindingProvider.PASSWORD_PROPERTY, getCurrentPassword() );
            context.put(SecurityConstants.ClientConstants.WSS_KEYSTORE_LOCATION,
"c:/mykeystore.jks");
            context.put(SecurityConstants.ClientConstants.WSS_KEYSTORE_PASSWORD,
"keystorepassword" );
            context.put(SecurityConstants.ClientConstants.WSS_KEYSTORE_TYPE, "JKS"
);
            context.put(SecurityConstants.ClientConstants.WSS_SIG_KEY_ALIAS, "your
signature alias" );
            context.put(SecurityConstants.ClientConstants.WSS_SIG_KEY_PASSWORD,
"your signature password" );
            context.put(SecurityConstants.ClientConstants.WSS_ENC_KEY_ALIAS, "your
encryption alias" );
            context.put(SecurityConstants.ClientConstants.WSS_ENC_KEY_PASSWORD,
"your encryption password" );
            System.out.println(proxy.myOperation("MyInput"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

In Example 11–9, the contents of *c:/dat/client-pdd.xml* referenced might be as follows:

```
! -- The contents of c:/dat/client-pdd.xml file mentioned above -- >
<oracle-webservice-clients>
  <webservice-client>
    <port-info>
      <policy-references>
        <policy-reference uri="management/Log_Msg_Policy" category="management"/>
        <policy-reference uri="oracle/wss10_username_token_with_message_
protection_client_policy" category="security"/>
      </policy-references>
    </port-info>
  </webservice-client>
</oracle-webservice-clients>
```

11.14 Configuring Local Optimization for a Policy

Oracle WSM supports a SOA local optimization feature for composite-to-composite invocations in which the reference of one composite specifies a Web service binding to a second composite running in the same container. Local optimization enables you to bypass the HTTP stack and SOAP/normalized message conversions during run time.

This SOA local optimization feature is described in "Policy Attachments and Local Optimization in Composite-to-Composite Invocations" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite* and summarized here.

11.14.1 Controlling When Local Optimization is Used

There are two ways to control the local optimization feature, and they have different scope:

- By adding the `oracle.webservices.local.optimization` property in the binding section of the `composite.xml` file. There are two possible values, `true` and `false`:
  - `true` -- Local optimization is used if the policy supports it as shown in Table 11–3 and the policy-level control is configured to use it as described in "Configuring the Policy-Level Optimization Control" on page 11-143.

If optimization is used, the policy is not applied.

– `false` -- Local optimization is not used, regardless of the how the policy-level control is configured and the default policy setting for the local-optimization property shown in Table 11–3.

  This setting forces the policy to be applied.

The composite-level property is independent of the policy-level configuration. That is, if you want to turn off the optimization regardless of whether a policy is attached, set the composite-level property to `false`.

See "Policy Attachments and Local Optimization in Composite-to-Composite Invocations" for information on overriding the local-optimization setting for a policy by adding the `oracle.webservices.local.optimization` property in the binding section of the `composite.xml` file.

- By configuring the optimization control for a policy, as described in "Configuring the Policy-Level Optimization Control" on page 11-143. The policy-level property controls the optimization wherever the policy is used, except as overridden by the composite-level property.

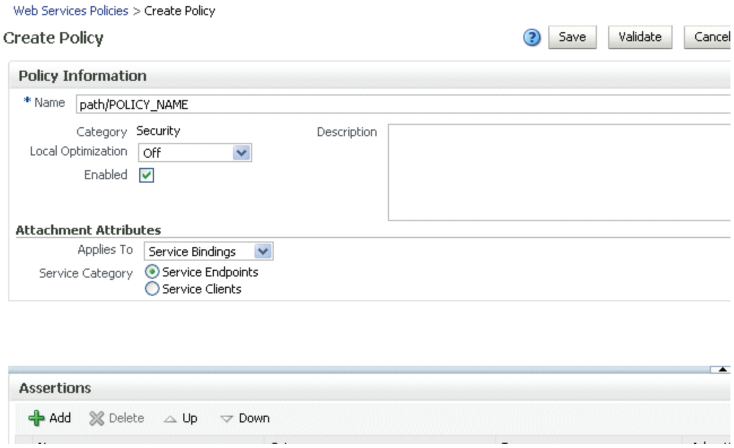
11.14.2 Configuring the Policy-Level Optimization Control

Notes: If there is a policy attached to the Web service, the policy may not be invoked if this optimization is used. Therefore, for each policy you need to decide whether you want to use the local optimization.

Oracle recommends that you do not change the optimization settings for the predefined policies because doing so may cause the policies to not be invoked, resulting in unexpected behavior.

The optimization control is available when you create or edit a policy, as shown in Figure 11–4.

Figure 11–4 Local Optimization Control When Creating a Policy



Web Services Policies > Create Policy

Create Policy

Policy Information

* Name: path/POLICY_NAME

Category: Security

Local Optimization: Off

Enabled: ☑

Description:

Attachment Attributes

Applies To: Service Bindings

Service Category: ● Service Endpoints ○ Service Clients

Assertions

There are three possible settings for the Local Optimization control: On, Off, and Check Identity:

- On -- Optimization is turned on and the policy is not applied.

- Off -- Optimization is turned off and the policy is applied. The request goes through the usual WS/SOAP/HTTP process.
- Check Identity -- Optimize only if a JAAS subject already exists in the current thread, indicating that authentication has already succeeded. Otherwise, go through the usual WS/SOAP/HTTP process.

Table 11–3 shows the predefined policies, and describes how each policy implements the local optimization feature.

Table 11–3 Default Optimization Setting of Predefined Policies

Policy Name	Default Optimization Setting
oracle/wsaddr10_policy	On
oracle/binding_authorization_denyall_policy	Always Off
oracle/binding_authorization_permitall_policy	Always Off
oracle/binding_permission_authorization_policy	Always Off
oracle/component_authorization_all_policy	Does not apply to bindings
oracle/log_policy	On
oracle/no_addressing_policy	Off
oracle/no_authentication_client_policy	Off
oracle/no_authentication_service_policy	Off
oracle/no_authorization_component_policy	Off
oracle/no_authorization_service_policy	Off
oracle/no_messageprotection_client_policy	Off
oracle/no_messageprotection_service_policy	Off
oracle/no_mtom_policy	Off
oracle/no_wsrm_policy	Off
oracle/sts_trust_config_client_policy	Off
oracle/sts_trust_config_service_policy	Off
oracle/whitelist_authorization_policy	Always Off
oracle/wsaddr_policy	On
oracle/wsmtom_policy	On

Table 11–3 (Cont.) Default Optimization Setting of Predefined Policies

Policy Name	Default Optimization Setting
oracle/wsrm10_policy	On
oracle/wsrm11_policy	On
oracle/wss_http_token_client_policy	Off
oracle/wss_http_token_service_policy	Off
oracle/wss_http_token_over_ssl_client_policy	Off
oracle/wss_http_token_over_ssl_service_policy	Off
oracle/wss11_kerberos_token_client_policy	Off
oracle/wss11_kerberos_token_service_policy	Off
oracle/wss_username_token_client_policy	Off
oracle/wss_username_token_service_policy	Off
oracle/wss_username_token_over_ssl_client_policy	Off
oracle/wss_username_token_over_ssl_service_policy	Off
oracle/wss10_message_protection_client_policy	On
oracle/wss10_message_protection_service_policy	On
oracle/wss10_username_token_with_message_protection_client_policy	Off
oracle/wss10_username_token_with_message_protection_service_policy	Off
oracle/wss10_x509_token_with_message_protection_client_policy	Off
oracle/wss10_x509_token_with_message_protection_service_policy	Off
oracle/wss10_saml_token_with_message_protection_client_policy	Check Identity
oracle/wss10_saml_token_with_message_protection_service_policy	Check Identity
oracle/wss11_saml_token_with_message_protection_client_policy	Check Identity

Table 11–3 (Cont.) Default Optimization Setting of Predefined Policies

Policy Name	Default Optimization Setting
oracle/wss11_saml_token_with_message_protection_service_policy	Check Identity
oracle/wss11_saml20_token_with_message_protection_client_policy	Check Identity
oracle/wss11_saml20_token_with_message_protection_service_policy	Check Identity
oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy	Off
oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy	Off
oracle/wss11_sts_issued_saml_with_message_protection_client_policy	Off
oracle/wss11_sts_issued_saml_with_message_protection_client_policy	Off
oracle/wss10_saml_token_with_message_integrity_client_policy	Check Identity
oracle/wss10_saml_token_with_message_integrity_service_policy	Check Identity
oracle/wss10_saml20_token_with_message_protection_client_policy	Check Identity
oracle/wss10_saml20_token_with_message_protection_service_policy	Check Identity
oracle/wss10_saml_token_client_policy	Check Identity
oracle/wss10_saml_token_service_policy	Check Identity
oracle/wss10_saml20_token_client_policy	Check Identity
oracle/wss10_saml20_token_service_policy	Check Identity
oracle/wss10_username_id_propagation_with_msg_protection_client_policy	Check Identity
oracle/wss10_username_id_propagation_with_msg_protection_service_policy	Check Identity
oracle/wss11_message_protection_client_policy	On

Table 11–3 (Cont.) Default Optimization Setting of Predefined Policies

Policy Name	Default Optimization Setting
oracle/wss11_message_protection_service_policy	On
oracle/wss11_username_token_with_message_protection_client_policy	Off
oracle/wss11_username_token_with_message_protection_service_policy	Off
oracle/wss11_x509_token_with_message_protection_client_policy	Off
oracle/wss11_x509_token_with_message_protection_service_policy	Off
oracle/wsrm10_policy	On
oracle/wsrm11_policy	On
oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy	Off
oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy	Off
oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy	Check Identity
oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy	Check Identity
wss11_saml_or_username_token_with_message_protection_client_policy	Check Identity
wss11_saml_or_username_token_with_message_protection_service_policy	Check Identity
wss11_saml_token_identity_switch_with_message_protection_client_policy	Off
wss10_saml_hok_token_with_message_protection_client_policy	Off
wss10_saml_hok_token_with_message_protection_service_policy	Off
oracle/wss_saml_or_username_token_over_ssl_service_policy	Check Identity
oracle/wss_saml_or_username_token_service_policy	Check Identity

Table 11–3 (Cont.) Default Optimization Setting of Predefined Policies

Policy Name	Default Optimization Setting
wss_saml_token_over_ssl_client_policy	Check Identity
wss_saml_token_over_ssl_service_policy	Check Identity
wss_saml20_token_over_ssl_client_policy	Check Identity
wss_saml20_token_over_ssl_service_policy	Check Identity
wss_saml_token_bearer_over_ssl_client_policy	Check Identity
wss_saml_token_bearer_over_ssl_service_policy	Check Identity
oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy	Off
oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy	Off
wss_saml20_token_bearer_over_ssl_client_policy	Check Identity
wss_saml20_token_bearer_over_ssl_service_policy	Check Identity
wss11_kerberos_token_with_message_protection_client_policy	Off
wss11_kerberos_token_with_message_protection_service_policy	Off
wss11_kerberos_token_with_message_protection_basic128_client_policy	Off
wss11_kerberos_token_with_message_protection_basic128_service_policy	Off

11.15 Configuring Fine-Grained Authorization Using Oracle Entitlements Server

As described in "Understanding Fine-Grained Authorization Using Oracle Entitlements Server" on page 10-118, Oracle Entitlements Server (OES) is a fine-grained authorization service you can use to secure applications and services end-to-end across the enterprise. OES is integrated with Oracle WSM, and you can use OES together with Oracle WSM for fine-grained authorization.

"Understanding Fine-Grained Authorization Using Oracle Entitlements Server" on page 10-118 describes the conceptual information you will need to configure OES integration, including a description of how resources are handled. That section also describes the division of labor: you configure the OES policies and Obligations from the OES console, and the Oracle WSM OES policies from Fusion Middleware Control, WLST, or a tool such as JDeveloper. If you have not already done so, read that section first.

This section describes how to configure OES integration, and includes the following topics:

- "Prerequisites for Configuring OES Integration" on page 11-149
- "Determine Attributes for Obligations" on page 11-149
- "Configure the OES Policies For Fine-Grained Authorization" on page 11-149
- "Configure the OES Policies For Coarse-Grained Authorization" on page 11-155
- "Configure the OES Policy For Masking" on page 11-157
- "Attach the Oracle WSM OES Policy" on page 11-160

11.15.1 Prerequisites for Configuring OES Integration

In addition to your Oracle WSM installation, you must also have an existing OES console configured, version 11.1.2.2.0 or later. OES must be installed on the same machine and to the same Oracle Middleware home as Oracle WSM.

OES is a part of the Oracle Identity and Access Management Suite, and is covered in the following documentation. This section assumes that you are already familiar with this content and with configuring and administering OES.

- See "Installing and Configuring Oracle Entitlements Server" in *Installation Guide for Oracle Identity and Access Management* for installation information.
- See *Administrator's Guide for Oracle Entitlements Server* for configuration and management information.

11.15.2 Determine Attributes for Obligations

As described in "How Attributes Are Processed" on page 10-127, OES allows you to create an Obligation in the OES console and provide multiple attribute name/value pairs. The attributes can be obtained from an XPath, an HTTP header, a message context, and constants (name/value), plus the set of static attributes (serviceURL, and so forth) that are always passed in authorization requests.

The easiest way to determine the information for the attributes is to deploy the application.   Then, examine the SOAP request or the WSDL and determine what attributes you want. There are two approaches:

- Deploy the application and use JDeveloper (or another mechanism) to look at the SOAP messages and determine what you need.
- Deploy the application and look at the WSDL of the deployed application to determine what you need.

  You can display the WSDL document for the web service endpoint as described in "Displaying the Web Service WSDL Document" on page 6-12.

11.15.3 Configure the OES Policies For Fine-Grained Authorization

As described in "OES Integration: The Big Picture" on page 10-119, there are two ways to contact OES for the authorization decision: a two-step (fine-grained) method and a single-step (coarse-grained) method. This section describes how to configure the OES policies for fine-grained authorization.

You specify which method to use via the `use.single.step` attribute in the `oracle/binding_oes_authorization_policy` and `oracle/component_oes_authorization_policy` policies when you later attach the Oracle WSM policy, as described in "Attach the Oracle WSM OES Policy" on page 11-160. However, you need to decide on the method you plan to use so that you can configure the OES authorization policy accordingly.

The two-step method is the more common scenario, and you therefore typically configure two OES authorization policies: one for defining Obligations and another for the actual authorization decision.

You use the OES console to create the basic artifacts (application, resource type, and so forth) and to add actions to the resource type and define the resource.

This section describes the following topics:

- "Configure the OES Resource" on page 11-150
- "Create Authorization Policy to Return Obligations" on page 11-151
- "Create the Actual OES Authorization Policy" on page 11-153

11.15.3.1 Configure the OES Resource

As described in "Resource Mapping and Naming" on page 10-125, you must map the OES resource name to the Oracle WSM resource name. When making an authorization call from Oracle WSM, the resource name is passed to OES, and this name must exactly match the one defined in the OES policy.

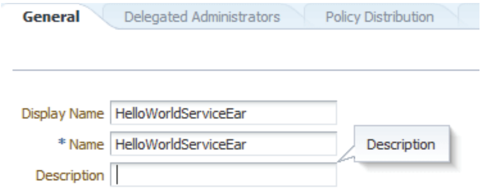
For the purpose of example, assume that you have a deployed SOAP web service with the following characteristics:

- The deployed application is named `HelloWorldServiceEar`.
- The resource type is `WS_SERVICE`.
- The web service name is `HelloWorldService`.
- The web service port name is `SayHelloPort`.
- The operations in the web service are `sayHello` and `sayHelloBytes`.

Perform the following steps to configure the OES resource:

1. Create the OES application name. The application name must match that of the deployed application. For example, `HelloWorldServiceEar`.

Figure 11–5 Create OES Application



Display Name: HelloWorldServiceEar
* Name: HelloWorldServiceEar
Description:

2. Create the OES resource type. The resource type must be `WS_SERVICE`.
3. Add actions for the `WS_SERVICE` resource type. The actions must be `request.lookup` and `authorize`.
4. Set the **Supports Resource Hierarchy** control.

By using the hierarchy, you can define the policy at the resource level or at the sub-resource (operation) level.

The Oracle WSM resource name includes the service name/port name/operation name. However, you do not have to define the resource to this granular level in the OES console. For example, you can define a policy with the service name that applies to all resources that start with that service. Or, you can define a policy for a resource name with the service name and port name and this policy would apply to all operations of that service.

Setting the **Supports Resource Hierarchy** control is described in "Creating a Resource Type" in *Administrator's Guide for Oracle Entitlements Server*.

5. Create the resource. The resource name can include the service name/port name/operation name.

11.15.3.2 Create Authorization Policy to Return Obligations

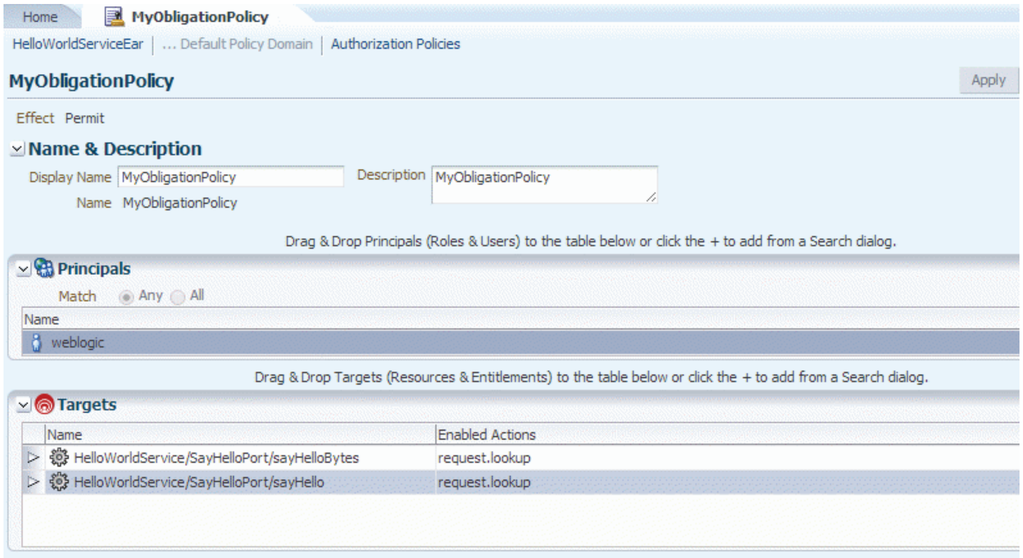
1. Create a new OES authorization policy for the application you created in "Configure the OES Resource" on page 11-150.

   Add the principals (roles and users) who should have access to the resource.

   Add the targets (with actions) to be protected by this policy.

   Figure 11–6 shows an example screen.

Figure 11–6 Adding Authorization Policy for Obligations


Home
MyObligationPolicy
HelloWorldServiceEar | ... Default Policy Domain | Authorization Policies
Apply

MyObligationPolicy

Effect Permit

Name & Description

Display Name: MyObligationPolicy

Description: MyObligationPolicy

Name: MyObligationPolicy

Drag & Drop Principals (Roles & Users) to the table below or click the + to add from a Search dialog.

Principals

Match ● Any ○ All

Name
weblogic

Drag & Drop Targets (Resources & Entitlements) to the table below or click the + to add from a Search dialog.

Targets

Name	Enabled Actions
HelloWorldService/SayHelloPort/sayHelloBytes	request.lookup
HelloWorldService/SayHelloPort/sayHello	request.lookup

2. From the Obligations tab, add Obligations.

   As described in "How Attributes Are Processed" on page 10-127, OES allows you to create an Obligation in the OES console and provide multiple attribute name/value pairs.

   The attributes can be obtained from an XPath, an HTTP header, a message context, and constants (name/value), plus the set of static attributes (serviceURL, and so forth) that are always passed in authorization requests. These attributes must follow a specific naming convention, as described in "How Attributes Are Processed" on page 10-127.

For the purpose of example, consider how Figure 11–7 is derived from and reflects the sample obligations shown in Table 11–4.

Remember that you must use these specific Obligation names, and case is insensitive. You can choose your own attribute names. The attribute values must match those of the request.

Note: The OES console requires that Obligation names be unique across an application. If you want to define more than one OES policy for the same application to return Obligations of the same type, you cannot use the same name. For example, if you add the `xpath` Obligation in `policy1`, you cannot also add it to `policy2` of the same application.

To make the Obligation name unique, use this naming convention:

```
<attribute_type>:<obligation_name>
```

where `attribute_type` is one of the supported types such as XPath, HTTPHeader or MessageContext. `obligation_name` can be any name to make it unique.

If an application has only a single OES policy to return Obligations and there is no chance of conflict, you can use the attribute type alone as the Obligation name.

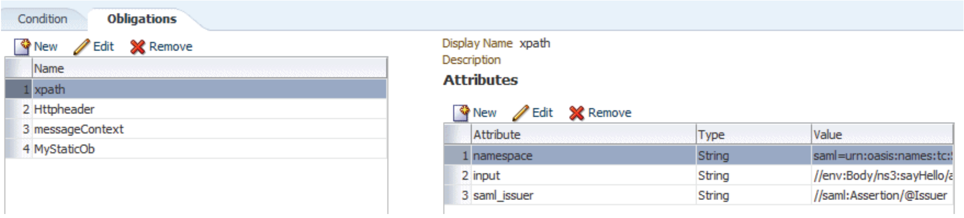
Table 11–4 Sample Obligations for the Policy

Obligation Name	Attribute Name	Attribute Value
xpath	input	//env:Envelope//env:Body/ns3:sayHello/arg0/text()

	namespace	saml=urn:oasis:names:tc:SAML:1.0:assertion
		env=http://schemas.xmlsoap.org/soap/envelope/
		ns3=http://helloworldservice.jaxws.wsmtest/
		Separate each attribute name with a comma (,). For example, `saml=... ,env=.. ,ns3=.`

	saml_issuer	//saml:Assertion/@Issuer
`Httpheader`	proxy_auth	Proxy-Authorization
	authHeader	Authorization
`messageContext`	authMethod	oracle.wsm.internal.authentication.method
	endpoint	oracle.j2ee.ws.runtime.endpoint-url
`MyStaticOb`	org	oracle
	country	US

Figure 11–7 Sample Obligations For Authorization Policy



	Name
1	xpath
2	Httpheader
3	messageContext
4	MyStaticOb

Display Name xpath

Description

Attributes

	Attribute	Type	Value
1	namespace	String	saml=urn:oasis:names:tc:
2	input	String	//env:Body/ns3:sayHello/
3	saml_issuer	String	//saml:Assertion/@Issuer

11.15.3.3 Create the Actual OES Authorization Policy

Create the OES authorization policy to perform the actual authorization.

This policy uses the Obligations provided by Oracle WSM to make the real authorization decision.

In addition, Oracle WSM passes to OES the authenticated subject, the target resource and requested action, as well as a set of implicit attributes (as described in Table 10–9, " Attribute Types Supported for OES Policies") that are always passed in authorization requests. Your authorization policy can use these values in the authorization decision. You can also use OES predefined attributes such as time, date, and so forth.

Perform the following steps to create the OES authorization policy for the actual authorization:

1. Add the attributes that you plan to use in your authorization condition. They must match the Obligations you created in "Create Authorization Policy to Return Obligations" on page 11-151.

   For example, to match the Obligations you created in "Create Authorization Policy to Return Obligations" on page 11-151 you would need to add the following attributes:

   - saml_issuer
   - input
   - authHeader
   - endpoint
   - country

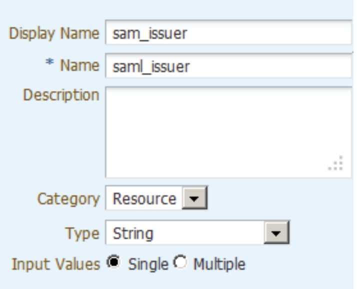
   Notes: If you plan to use any of the implicit attributes (as described in Table 10–9, " Attribute Types Supported for OES Policies"), you must add them as well.

   All Attributes and Functions (both custom and predefined) are created, collected and further managed under the Extensions node of the Application. For more information, see "Managing Attributes and Functions as Extensions" in *Administrator's Guide for Oracle Entitlements Server*.

   In the OES navigation pane, expand the application for which you are creating the authorization policy. Expand **Extensions**, then select **Attributes** and click the icon to create the new attributes.

   You can choose **Save and create another** from the drop-down control to create multiple attributes from the single page.

Figure 11–8 Create New Attributes



Display Name: sam_issuer

* Name: saml_issuer

Description:

Category: Resource

Type: String

Input Values: ● Single ○ Multiple

2. Add all of the new attributes to your resource type.

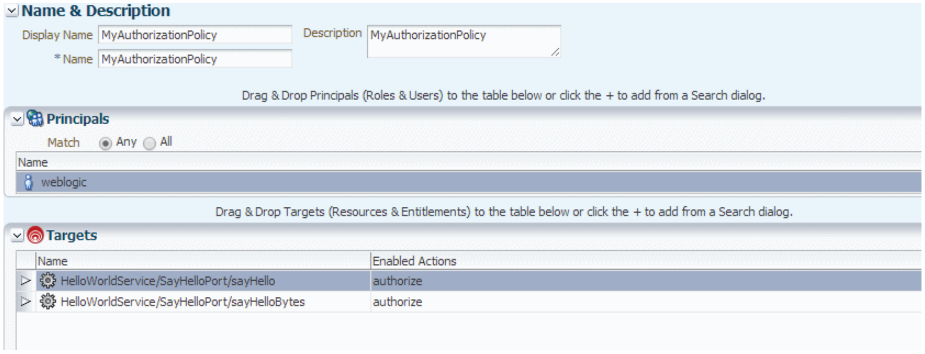
3. Create a new OES authorization policy for the application you created in "Configure the OES Resource" on page 11-150.

   Add the principals (roles and users) who should have access to the resource.

   Add the targets (with actions) to be protected by this policy.

   Figure 11–9 shows an example screen. The policy is set up to evaluate the authorization of user `weblogic` upon access to the `sayHello` and `sayHelloBytes` resources of `HelloWorldService`.

Figure 11–9 Adding Actual Authorization Policy



Name & Description

Display Name: MyAuthorizationPolicy

*Name: MyAuthorizationPolicy

Description: MyAuthorizationPolicy

Drag & Drop Principals (Roles & Users) to the table below or click the + to add from a Search dialog.

Principals

Match ● Any ○ All

Name
weblogic

Drag & Drop Targets (Resources & Entitlements) to the table below or click the + to add from a Search dialog.

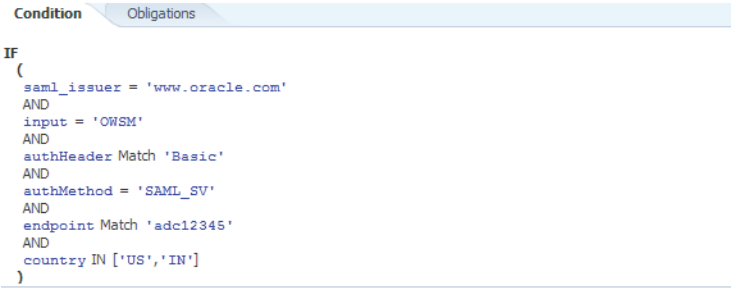
Targets

Name	Enabled Actions
HelloWorldService/SayHelloPort/sayHello	authorize
HelloWorldService/SayHelloPort/sayHelloBytes	authorize

4. From the Conditions tab for this policy, click **Edit** to create a new condition.

   Complete the condition based on the attributes you added in Step 1, as shown in Figure 11–10.

Figure 11–10 Create the Condition



```
IF
 (
  saml_issuer = 'www.oracle.com'
  AND
  input = 'OWSM'
  AND
  authHeader Match 'Basic'
  AND
  authMethod = 'SAML_SV'
  AND
  endpoint Match 'adc12345'
  AND
  country IN ['US','IN']
 )
```

In this example, the policy returns PERMIT if:

- The user is `weblogic`.

- The resource is `HelloWorldServiceEar/ HelloWorldService/SayHelloPort`.
- The SAML issuer is `www.oracle.com`.
- The request SOAP message has "Oracle WSM" at `//env:Body/ns3:sayHello/arg0`.
- Other conditions are met.

5. Attach the Oracle WSM OES policy, as described in "Attach the Oracle WSM OES Policy" on page 11-160.

11.15.4 Configure the OES Policies For Coarse-Grained Authorization

As described in "OES Integration: The Big Picture" on page 10-119, there are two ways to contact OES for the authorization decision: a two-step (fine-grained) method and a single-step (coarse-grained) method. This section describes how to configure the OES policies for coarse-grained authorization.

You specify which method to use via the `use.single.step` attribute in the `oracle/binding_oes_authorization_policy` and `oracle/component_oes_authorization_policy` policies when you later attach the Oracle WSM policy, as described in "Attach the Oracle WSM OES Policy" on page 11-160. However, you need to decide on the method you plan to use so that you can configure the OES authorization policy accordingly.

This section describes the following topics for coarse-grained authorization:

- "Configure the OES Resource" on page 11-155
- "Create the Actual OES Authorization Policy" on page 11-156

11.15.4.1 Configure the OES Resource

As described in "Resource Mapping and Naming" on page 10-125, you must map the OES resource name to the Oracle WSM resource name. When making an authorization call from Oracle WSM, the resource name is passed to OES, and this name must exactly match the one defined in the OES policy.

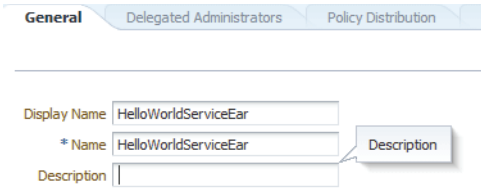
For the purpose of this example, assume that you have a deployed SOAP web service with the following characteristics:

- The deployed application is named `HelloWorldServiceEar`.
- The resource type is always `WS_SERVICE`.
- The web service name is `HelloWorldService`.
- The web service port name is `SayHelloPort`.
- The operations in the web service are `sayHello` and `sayHelloBytes`.

Perform the following steps to configure the OES resource:

1. Create the OES application name. The application name must match that of the deployed application. For example, `HelloWorldServiceEar`.

Figure 11–11 Create OES Application



Display Name: HelloWorldServiceEar

* Name: HelloWorldServiceEar

Description:

2. Create the OES resource type. The resource type must be `WS_SERVICE`.
3. Add the action for the resource type. The action must be `authorize`.
4. Set the **Supports Resource Hierarchy** control.

   By using the hierarchy, you can define the policy at the resource level or at the sub-resource (operation) level.

   The Oracle WSM resource name includes the service name/port name/operation name. However, you do not have to define the resource to this granular level in the OES console. For example, you can define a policy with the service name that applies to all resources that start with that service. Or, you can define a policy for a resource name with the service name and port name and this policy would apply to all operations of that service.

   Setting the **Supports Resource Hierarchy** control is described in "Creating a Resource Type" in *Administrator's Guide for Oracle Entitlements Server*.
5. Create the resource. The resource name must be of the form `service name/port name/operation name`, depending on how you want to utilize the resource hierarchy.

11.15.4.2 Create the Actual OES Authorization Policy

Create the OES authorization policy to perform the actual authorization.

Oracle WSM passes to OES the authenticated subject, the target resource and requested action, as well as a set of implicit attributes (as described in Table 10–9, " Attribute Types Supported for OES Policies") that are always passed in authorization requests.

Your authorization policy can use these values in the authorization decision. You can also use OES predefined attributes such as time, date, and so forth.

Perform the following steps to create the OES authorization policy for the actual authorization:

1. If you plan to use any of the implicit attributes (as described in Table 10–9, " Attribute Types Supported for OES Policies"), you must add them.

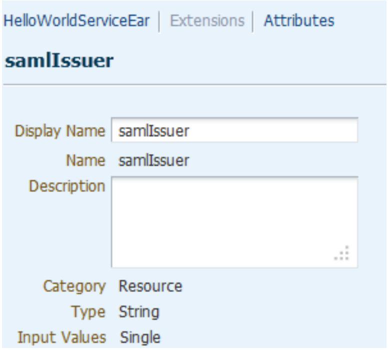
   > **Note:** All Attributes and Functions (both custom and predefined) are created, collected and further managed under the Extensions node of the Application. For more information, see "Managing Attributes and Functions as Extensions" in *Administrator's Guide for Oracle Entitlements Server*.

   In the OES navigation pane, expand the application for which you are creating the authorization policy. Expand **Extensions**, then select **Attributes** and click the icon to create the new attributes.

You can choose **Save and create another** from the drop-down control to create multiple attributes from the single page.

2. Add all of the implicit attributes you plan to use to your resource type, as shown in Figure 11–12.

Figure 11–12 Add Implicit Attributes Needed for Conditions



HelloWorldServiceEar | Extensions | Attributes

samlIssuer

Display Name	samlIssuer
Name	samlIssuer
Description	
Category	Resource
Type	String
Input Values	Single

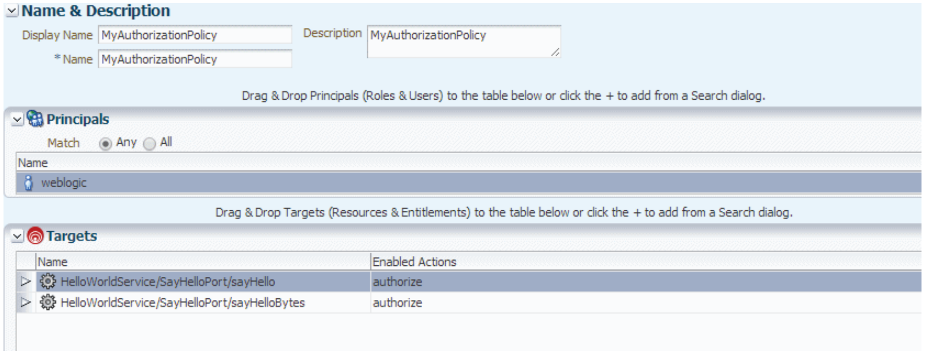
3. Create a new OES authorization policy for the application you created in "Configure the OES Resource" on page 11-155.

   Add the principals (roles and users) who should have access to the resource.

   Add the targets (with actions) to be protected by this policy.

   Figure 11–13 shows an example screen.

Figure 11–13 Adding Actual Authorization Policy



Name & Description

Display Name	MyAuthorizationPolicy	Description	MyAuthorizationPolicy
* Name	MyAuthorizationPolicy		

Drag & Drop Principals (Roles & Users) to the table below or click the + to add from a Search dialog.

Principals

Match ● Any ○ All

Name
weblogic

Drag & Drop Targets (Resources & Entitlements) to the table below or click the + to add from a Search dialog.

Targets

Name	Enabled Actions
HelloWorldService/SayHelloPort/sayHello	authorize
HelloWorldService/SayHelloPort/sayHelloBytes	authorize

4. Attach the Oracle WSM OES policy, as described in "Attach the Oracle WSM OES Policy" on page 11-160.

11.15.5 Configure the OES Policy For Masking

This section describes the following topics:

- "Configure the OES Resource" on page 11-158
- "Create Masking Policy to Return Obligations" on page 11-158
- "Create the Actual OES Masking Policy" on page 11-159

11.15.5.1 Configure the OES Resource

As described in "Resource Mapping and Naming" on page 10-125, you must map the OES resource name to the Oracle WSM resource name. When making a masking call from Oracle WSM, the resource name is passed to OES, and this name must exactly match the one defined in the OES policy.

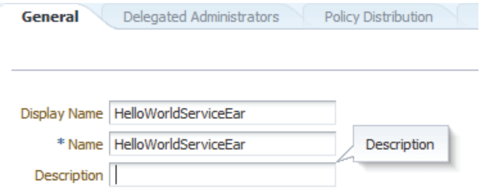
For the purpose of example, assume that you have a deployed SOAP web service with the following characteristics:

- The deployed application is named `HelloWorldServiceEar`.
- The resource type is always `WS_SERVICE`.
- The web service name is `HelloWorldService`.
- The web service port name is `SayHelloPort`.
- The operations in the web service are `sayHello` and `sayHelloBytes`.

Perform the following steps to configure the OES resource:

1. If you have not already done so, create the OES application name. The application name must match that of the deployed application. For example, `HelloWorldServiceEar`.

Figure 11–14 Create OES Application



Display Name: HelloWorldServiceEar
* Name: HelloWorldServiceEar
Description:

2. If you have not already done so, create the OES resource type. The resource type must be `WS_SERVICE`.
3. If you have not already done so, add actions for the resource type. The actions must be `response.lookup` and `mask`.
4. If you have not already done so, set the **Supports Resource Hierarchy** control.

   Setting the **Supports Resource Hierarchy** control is described in "Creating a Resource Type" in *Administrator's Guide for Oracle Entitlements Server*.
5. If you have not already done so, create the resource. The resource name must be of the form `service name/port name/operation name`, depending on how you want to utilize the resource hierarchy.

11.15.5.2 Create Masking Policy to Return Obligations

1. Create a new OES policy for the application you created in "Configure the OES Resource" on page 11-158.

   Add the principals (roles and users) who should have access to the resource.

   Add the targets (with action `response.lookup`) to be protected by this policy.
2. From the Obligations tab, add Obligations.

   As described in "How Attributes Are Processed" on page 10-127, OES allows you to create an Obligation in the OES console and provide multiple attribute name/value pairs.

For this masking use case, create a set of XPath queries to get the attributes you may want to mask.

As a result of these Obligations, Oracle WSM gets all of defined attributes and Xpath queries and runs them on the current response SOAP message. (If nothing is returned in this call then execution stops and no masking is performed.) Oracle WSM uses the result of this query to call the actual masking policy described in "Create the Actual OES Masking Policy" on page 11-159.

You use the attribute names from this Obligation when you create the Obligation for the actual masking policy.

11.15.5.3 Create the Actual OES Masking Policy

Create the OES authorization policy to perform the actual masking.

This policy uses the Obligations provided by Oracle WSM to make the real masking decision.

Perform the following steps to create the OES policy for the actual masking:

1. Add the attributes that you plan to use in your condition.

   If you plan to use any of the implicit attributes (as described in Table 10–9, " Attribute Types Supported for OES Policies"), you must add them as well.

   > **Note:** All Attributes and Functions (both custom and predefined) are created, collected and further managed under the Extensions node of the Application. For more information, see "Managing Attributes and Functions as Extensions" in *Administrator's Guide for Oracle Entitlements Server*.

   In the OES navigation pane, expand the application for which you are creating the authorization policy. Expand **Extensions**, then select **Attributes** and click the icon to create the new attributes.

   You can choose **Save and create another** from the drop-down control to create multiple attributes from the single page.

2. Add all of the new attributes to your resource type.

3. Create a new OES authorization policy for the application you created in "Configure the OES Resource" on page 11-158.

   Add the principals (roles and users) who should have access to the resource.

   Add the targets (with action `mask`) to be protected by this policy.

4. From the Obligations tab, add Obligations.

   These Obligations specify whether the attribute should be passed as-is or masked. Oracle WSM honors the Obligation returned by OES and masks attributes marked sensitive by OES.

   Return the value with which you want to replace an attribute.

   For example, if in the obligation policy you added an XPath as `name=//env:Envelope//env:Body/ns3:sayHelloResponse/return/text()`, in the masking policy you might add the Obligation in an XPath as `name=xxxxx` or `name=****`, where `name` matches in both policies.

5. Attach the Oracle WSM OES policy, as described in "Attach the Oracle WSM OES Policy" on page 11-160.

11.15.6 Attach the Oracle WSM OES Policy

Make a copy of the preconfigured oracle/binding_oes_authorization_policy, oracle/component_oes_authorization_policy or oracle/binding_oes_masking_policy and then attach the copy to your web service. Perform the following steps:

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure OES integration. Select the domain.
2. In the content pane, click **WebLogic Domain**, then **Web Services**, and then **Policies**.
3. Select the `oracle/binding_oes_authorization_policy`, `oracle/component_oes_authorization_policy` or `oracle/binding_oes_masking_policy` policy and make a copy.
4. Edit the attributes of the copy. (You can instead choose to override the attributes, as described in a subsequent step.)

   The `use.single.step` attribute controls the way in which you contact OES for the authorization decision: a two-step (fine-grained) method and a single-step (coarse-grained) method. The default is false, which uses the two-step (fine-grained) method.   In the one-step method you do not use Obligations but you can use the implicit attributes in the conditions.

   You can set this attribute in the `oracle/binding_oes_authorization_policy` and `oracle/component_oes_authorization_policy` policies. The `oracle/binding_oes_masking_policy` is always two-step.

   The **OES Based Authorization** setting uses the guard element (see orawsp:guard) to define resource, action, and constraint match values. These values allow the assertion execution only if the result of the guard is true. That is, if the accessed resource name and action match, only then is the assertion allowed to execute. By default, resource name and action use the wildcard asterisk "*" and everything is allowed.

   If you followed the resource naming convention when creating the OES policy, you do not need to explicitly set the resource name.
5. Attach the policy at design time or post-deployment, as described in Chapter 8, "Attaching Policies to Web Services".

   Attach the Oracle WSM `oracle/binding_oes_authorization_policy` or `oracle/component_oes_authorization_policy` policy, alone or combination with the `oracle/binding_oes_masking_policy` policy.
6. Optionally, use Fusion Middleware Control, WLST, or JDeveloper (or another mechanism) to override attributes. If you followed the resource naming convention when creating the OES policy, overriding resource properties is not required.

   See "Configuration Properties and Overrides" on page 11-161 for a description of the attributes you can override.

   See "Attaching Web Service Policies Permitting Overrides" on page 8-25 for information on overriding configuration policies.
7. Also attach any of the authentication policies at design time or post-deployment, as described in Chapter 8, "Attaching Policies to Web Services".

11.15.6.1 Configuration Properties and Overrides

You can set the configuration properties shown in Table 11–5 for the policies when you attach the policy, or override them.

If you followed the resource naming convention when creating the OES policy, the resource names are derived and overriding the properties is not required.

If you override some but not all values, the remainder are derived.

Table 11–5 Oracle WSM OES Configuration Properties

Name	Description
`application.name`	The deployed application name defined in OES. (For SOA, the composite name is used as the application name.) Value can be static or dynamic that uses ${} notation.
`resource.type`	Resource type defined in OES. Value can be static or dynamic that uses ${} notation. ■ For SOAP application, must be `WS_SERVICE`. ■ For SOA component, must be `COMPONENT`.
`resource.name`	Resource name defined in OES. Value can be static or dynamic that uses ${} notation. ■ For SOAP and SOA reference, must be of the form `web-service-name/port/web service operation`. ■ For SOA component, must be of the form `SOA component name/web service operation`.
`lookup.action`	Action that will be used during attributes lookup. Can be `request.lookup` or `response.lookup`. Value can be static or dynamic that uses ${} notation.
`execute.action`	Action that will be used during real authorization or masking. Default values are `authorize` for authorization and `mask` for masking use case. Value can be static or dynamic that uses ${} notation.
`use.single.step`	Set value to true to skip lookup phase. Does not apply to masking policy.
`reference.prioroty`	Optional property that specifies the priority of the policy attachment.

These properties allow both static and dynamic values. Dynamic values use one or more `${}` operators and allow separator characters such as a period or slash. For example, if you specify the value of `resource.name` as `${PORT}/${OPERATION}`, then it could resolve to `myPort/operation`. As another example, `${MODULE}.${SERVICE}` could resolve to `myModule.myService`.

The possible dynamic values are as follows:

- APPLICATION
- MODULE
- SERVICE (For SOAP and SOA reference, it is WSDL web service name.
- PORT (WSDL service port name)
- OPERATION (web service operation for SOAP.)
- COMPOSITE (SOA composite name)

- COMPONENT (SOA component name)
- NAMESPACE

12 Testing Web Services

This chapter includes the following sections:

12.1 Testing Your Web Services

This section describes how to use the Fusion Middleware Control Test Web Service page to verify that you are receiving the expected results from the Web service.

The Test Web Service page allows you to test any of the operations exposed by a Web service. You can test Web services that are deployed on any accessible host; the Web service does not have to be deployed on this host.

> **Note:** The Test Web Service page can parse WSDL URLs that contain ASCII characters only. If the URL contains non-ASCII characters, the parse operation fails. To test a Web service that has non-ASCII characters in the URL, allow your browser to convert the WSDL URL and use the resulting encoded WSDL URL in the Test Web Service page.
>
> When testing Web services that use policies, the Oracle WSM component must be installed in the same domain from which Fusion Middleware Control is being run. Otherwise, an invalid policy exception will be returned.

You can navigate to the Test Web Service page in many ways. This section describes two typical ways to do so.

To test your Web service

1. Access the Test Web Service page using either of these typical ways to do so.
   - From the Oracle WebLogic Server Domain Home page:

   a. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to test a Web service.
   b. Select the domain.
   c. From the **WebLogic Domain** menu, select **Web Services**, and then **Test Web Service**. The Test Web Service input page appears.
   d. Enter the WSDL of the Web service you want to test. If you do not know the WSDL, click the search icon and select from the registered Web services, if any.
   e. Click **Parse WSDL**.

      If the WSDL is secured with HTTP Basic Authentication, click **HTTP Basic Auth Option for WSDL Access** and enter the username and password before parsing the WSDL.

      The complete Test Web Services page displays, as shown in Figure 12–1.

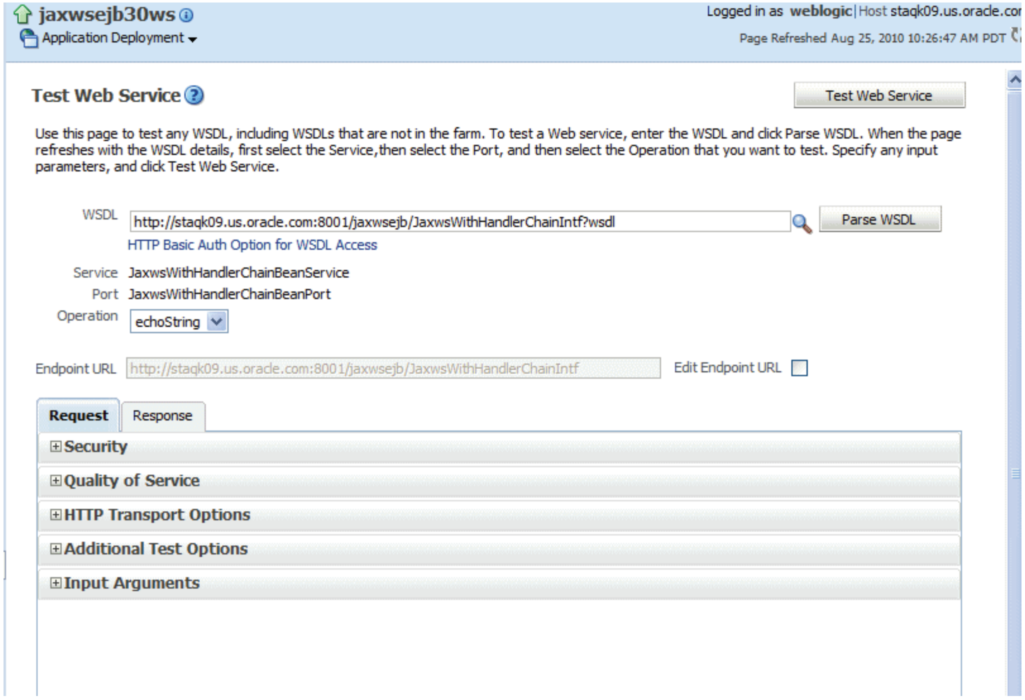
- From the Web Service Application Home page:
   a. In the navigator pane, expand **Application Deployments** to view the applications in the domain.
   b. Select the application for which you want to test the Web service.
   c. In the Web Services section of the page, click **Test** for the Web service endpoint you want to test.

      The Test Web Service page displays, as shown in Figure 12–1. Note that the WSDL field is automatically populated with the WSDL for the endpoint.
   d. Click **Parse WSDL**.

      If the WSDL is secured with HTTP Basic Authentication, click **HTTP Basic Auth Option for WSDL Access** and enter the username and password before parsing the WSDL.

The Test Web Service page is shown in Figure 12–1. Note that the test option sections are collapsed by default.

Figure 12–1 Test Web Service Page in Collapsed View



jaxwsejb30ws

Application Deployment

Logged in as weblogic | Host staqk09.us.oracle.cor

Page Refreshed Aug 25, 2010 10:26:47 AM PDT

Test Web Service

Use this page to test any WSDL, including WSDLs that are not in the farm. To test a Web service, enter the WSDL and click Parse WSDL. When the page refreshes with the WSDL details, first select the Service,then select the Port, and then select the Operation that you want to test. Specify any input parameters, and click Test Web Service.

WSDL http://staqk09.us.oracle.com:8001/jaxwsejb/JaxwsWithHandlerChainIntf?wsdl

HTTP Basic Auth Option for WSDL Access

Service JaxwsWithHandlerChainBeanService

Port JaxwsWithHandlerChainBeanPort

Operation echoString

Endpoint URL http://staqk09.us.oracle.com:8001/jaxwsejb/JaxwsWithHandlerChainIntf

Edit Endpoint URL □

2. Select the service and port to be tested. If the WSDL has multiple services and ports, these fields are available as drop-down menus. If the WSDL has only one service and port, these fields are read-only, as shown in Figure 12–1.

3. Select the operation that you want to test from the Operation menu. The available operations are determined from the WSDL.

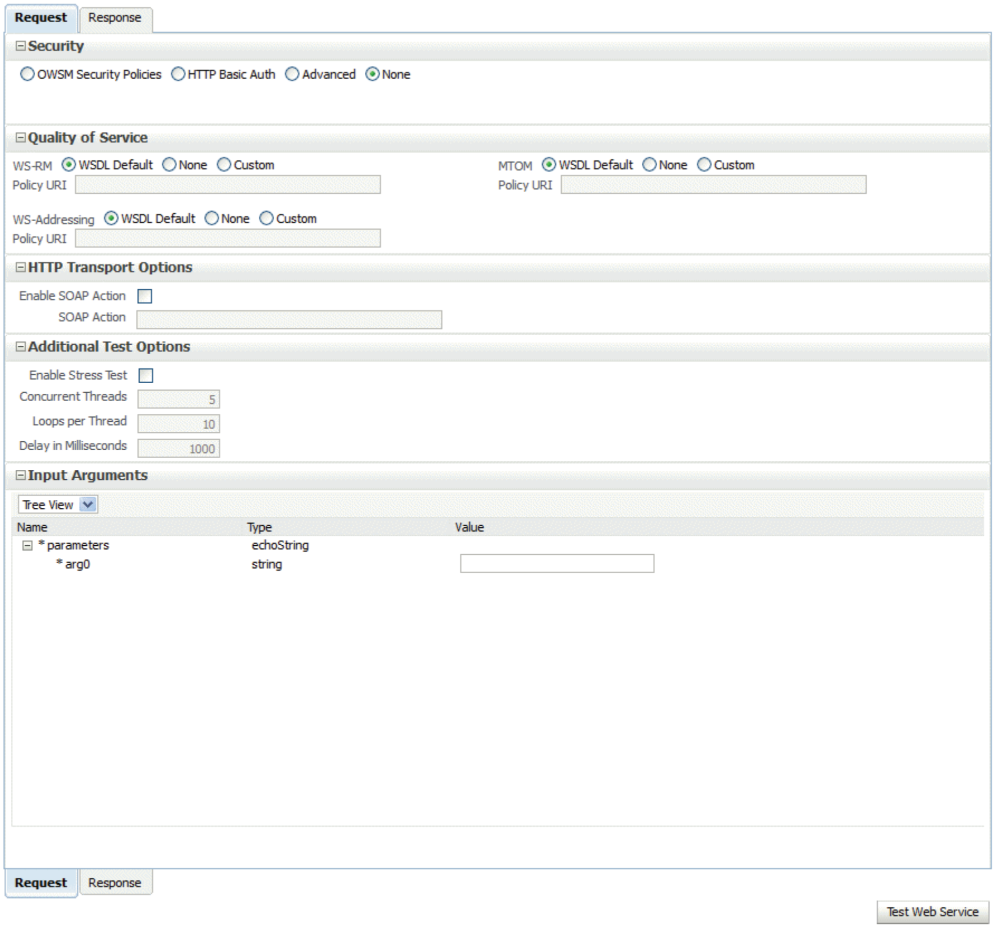
To test a RESTful Web service, select the GET or POST service port operations.

4. If you want to change the endpoint URL of the test, click **Edit Endpoint URL** and make the change.

5. Select the **Request** tab if it is not already selected.

6. Expand the test option sections by clicking the plus sign (+) next to the section name. The expanded view of the Test Web Service page is shown in Figure 12–2.

Figure 12–2 Bottom Portion of Test Web Service Page in Expanded View



⊟ **Security**

○ OWSM Security Policies ○ HTTP Basic Auth ○ Advanced ● None

⊟ **Quality of Service**

WS-RM ● WSDL Default ○ None ○ Custom
Policy URI

MTOM ● WSDL Default ○ None ○ Custom
Policy URI

WS-Addressing ● WSDL Default ○ None ○ Custom
Policy URI

⊟ **HTTP Transport Options**

Enable SOAP Action □
SOAP Action

⊟ **Additional Test Options**

Enable Stress Test □
Concurrent Threads 5
Loops per Thread 10
Delay in Milliseconds 1000

⊟ **Input Arguments**

Name	Type	Value
⊟ * parameters	echoString	
* arg0	string	

7. In the Security section, specify whether you want to test a Web service using Oracle WSM security policies, basic HTTP authentication, authentication from a custom policy, or no credentials. The security setting is not determined from a policy in the WSDL; you can specify the type of security you want to test. The default is **None**. Depending on the option selected, additional fields are displayed. For details about the options available, see "Enabling Security Testing" on page 12-6.

   When testing RESTful Web services, because the SOAP protocol is not used, the only security options are **HTTP Basic Authentication** or **None**.

8. In the Quality of Service section, specify whether you want to explicitly test a Reliable Messaging (WS-RM), WS-Addressing, or MTOM policy. For details about the options available, see "Enabling Quality of Service Testing" on page 12-9.

   Note: This section is not available when testing RESTful Web services.

9. In the HTTP Transport section, the test mechanism uses the WSDL to determine whether a SOAP action is available to test. If available, specify whether you want send the request with the SOAP action HTTP header. For more information, see "Enabling HTTP Transport Options" on page 12-10.

Note: This section is not available when testing RESTful Web services.

10. In the Additional Test Options section, select the **Enable Stress Test** option if you want to invoke the Web service multiple times simultaneously. If you select this option, you can also provide values for the stress test options, or accept the defaults. For more information, see "Stress Testing the Web Service Operation" on page 12-10.

11. In the Input Arguments section, enter the input arguments for the Web service in the **Value** fields. The parameters and type, and the required input values, are determined from the WSDL.

    Select Tree View or XML View to toggle between a hierarchical list of input parameters and the XML content.

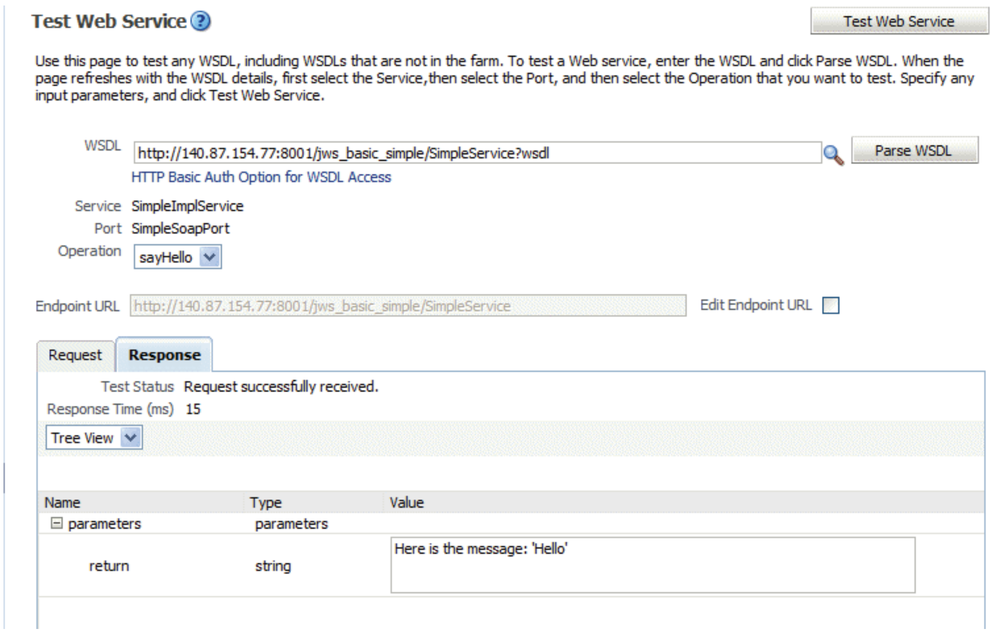
12. Click **Test Web Service** to initiate the test.

    The test results appear in the **Response** tab upon completion.

    If the test is successful, the **Test Status** field indicates *Request Successfully received* and the response time is displayed, as shown in Figure 12–3.

Note: When running SOA composite tests, the Response tab will indicate whether a new composite was generated. You can also click the **Launch Flow Trace** button to open the Flow Trace window, where you can view the flow of the message through various composite and component instances.

Figure 12–3 Successful Test



Test Web Service

Use this page to test any WSDL, including WSDLs that are not in the farm. To test a Web service, enter the WSDL and click Parse WSDL. When the page refreshes with the WSDL details, first select the Service,then select the Port, and then select the Operation that you want to test. Specify any input parameters, and click Test Web Service.

WSDL http://140.87.154.77:8001/jws_basic_simple/SimpleService?wsdl

Service SimpleImplService

Port SimpleSoapPort

Operation sayHello

Endpoint URL http://140.87.154.77:8001/jws_basic_simple/SimpleService

Test Status Request successfully received.

Response Time (ms) 15

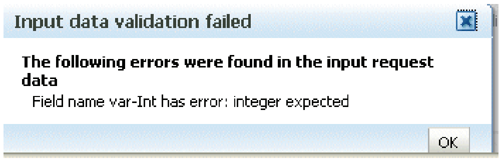
Tree View

Name	Type	Value
⊟ parameters	parameters	
return	string	Here is the message: 'Hello'

If the test fails, an error message is displayed. For example, Figure 12–4 shows an error resulting from a type error in the *var-Int* parameter. In this particular instance, *string* data was entered when an *int* was expected.

Note: The results on the **Response** tab are a simplified version of the standard Web service results.

Figure 12–4 Data Validation Error



Input data validation failed

The following errors were found in the input request data

Field name var-Int has error: integer expected

12.2 Editing the Input Arguments as XML Source

You can view the input arguments in a user-friendly form, or you can edit the XML source code directly. If you edit the XML source directly, you must enter valid XML. Use the drop-down list in the Input Arguments section of the page to toggle between **Tree View** and **XML View.**

12.3 Enabling Security Testing

You can use the Test Web Service Page to test Web services security using Oracle WSM security policies, HTTP basic authentication, or custom policies. You can choose the type of test by selecting one of the options in the Security section of the page.

The security setting is not determined from a policy in the WSDL; you can specify the type of security you want to test. The default is **None**.

The following options available, and described in more detail in the subsequent sections:

- **OWSM Security Policies**– Uses the credentials and other security options required by the Oracle WSM security policies for authentication and message protection.
- **HTTP Basic Auth** – Inserts the username and password credentials in the HTTP transport header. Both the username and password are required.

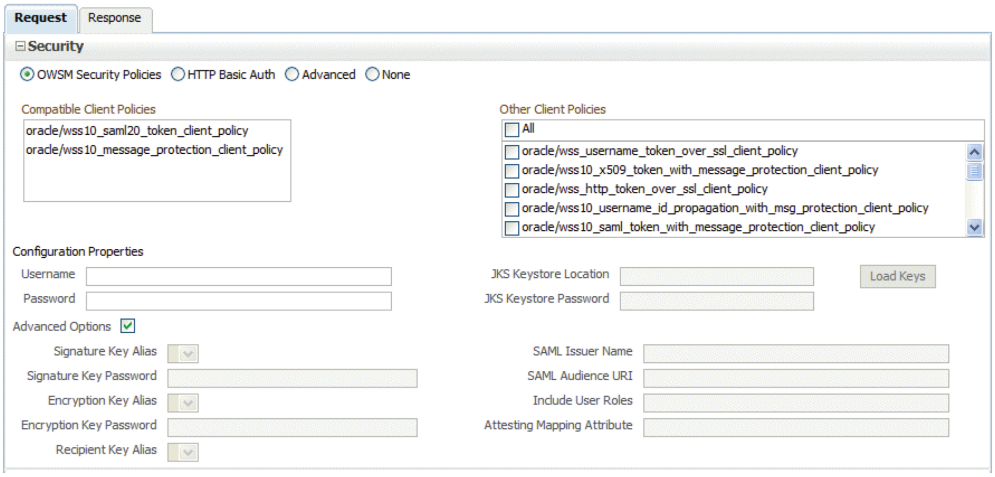
  If you do specify a username and password, they must exist and be valid.)
- **Advanced** – Uses a custom policy to authenticate the user. You must specify the URI for the policy. You can also specify configuration overrides.
- **None** – No credentials are included.

Note: When testing RESTful Web services, because the SOAP protocol is not used, the only security options are **HTTP Basic Authentication** or **None**.

OWSM Security Policies

The options available when you select **OWSM Security Policies**, are shown in Figure 12–5. Note that in this figure the **Advanced Options** field is selected to display all of the available fields.

Figure 12–5 OWSM Security Policies Test Options



Request | Response

⊟ Security

● OWSM Security Policies ○ HTTP Basic Auth ○ Advanced ○ None

Compatible Client Policies

oracle/wss10_saml20_token_client_policy
oracle/wss10_message_protection_client_policy

Other Client Policies

□ All
□ oracle/wss_username_token_over_ssl_client_policy
□ oracle/wss10_x509_token_with_message_protection_client_policy
□ oracle/wss_http_token_over_ssl_client_policy
□ oracle/wss10_username_id_propagation_with_msg_protection_client_policy
□ oracle/wss10_saml_token_with_message_protection_client_policy

Configuration Properties

Username
Password
JKS Keystore Location
JKS Keystore Password
Load Keys

Advanced Options ☑

Signature Key Alias
Signature Key Password
Encryption Key Alias
Encryption Key Password
Recipient Key Alias
SAML Issuer Name
SAML Audience URI
Include User Roles
Attesting Mapping Attribute

To test the Web service security using Oracle WSM security policies:

1. From the **Compatible Client Policies** list, which displays the compatible client policies as specified in the WSDL, select the client policy to test.

   Alternatively, to perform a negative test on the endpoint, you can select a non-compatible policy, or **All** from the **Other Client Policies** list.

   > **Notes:** Non-security policies and policies not supported by the test function are shown as read-only and cannot be selected. For a list of client policies supported by the test function, see "Supported Client Security Policies" on page 12-8.
   >
   > Service polices that support multiple client security policies are not supported by the test function.

   The Configuration Properties required by the selected policy are indicated with an asterisk (*). For example:

   - For username_token and http_token policies, the **Username** and **Password** fields are required; for SAML policies only the **Username** field is required.
   - For message protection and two-way SSL policies, the **JKS Keystore Location** and **JKS Keystore Password** fields are required. (These fields are not required to be set for one-way SSL policies.)

2. Provide values for the required fields as determined by the policy.

   If the **JKS Keystore Location** and **JKS Keystore Password** fields are required by the policy, enter the location and password of a temporary user-created keystore that is NFS accessible and click **Load Keys**. The associated keystore fields under **Advanced Options** are populated with the aliases specified in the keystore.

   For more information about creating a keystore, see "Generating Private Keys and Creating the Java Keystore" on page 10-9.

3. Click **Advanced Options**. Additional keystore alias and SAML properties fields are displayed. Properties required by the selected policies are indicated with an asterisk. Enter the required values, or provide override values in the applicable fields.

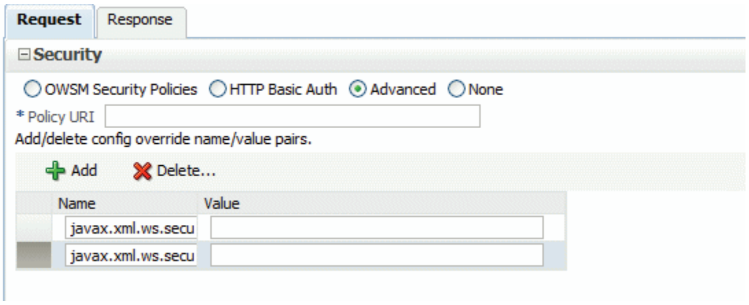
HTTP Basic Auth

This option requires **Username** and **Password** credentials that are inserted into the HTTP transport header. The username and password must exist and be valid for the WebLogic Server.

Advanced

The options available when you select the **Advanced** option are shown in Figure 12–6.

Figure 12–6 Advanced Test Options



○ OWSM Security Policies ○ HTTP Basic Auth ● Advanced ○ None

* Policy URI

Add/delete config override name/value pairs.

Name	Value
javax.xml.ws.secu	
javax.xml.ws.secu	

The Advanced option allows you to use a custom policy to test the Web service security. To do so:

1. Specify the URI of the custom policy in the **Policy URI** field. This field is required.
2. Specify any configuration overrides for the policy in the **Name** and **Value** fields. To add properties, click **Add** and provide the name/value pair for the configuration override. To delete a property, select it in the table and click **Delete**.

> **Note:** Properties must be specified using the full name of each property, for example `oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_LOCATION`. For a complete list of property names for overridable properties, see Table 11–2 in "Using Client Programmatic Configuration Overrides" on page 11-128.

12.3.1 Supported Client Security Policies

The following Oracle WSM client security policies are supported by the test function:

- oracle/wss_http_token_client_policy
- oracle/wss_http_token_over_ssl_client_policy
- oracle/wss_saml_token_bearer_over_ssl_client_policy
- oracle/wss_saml_token_over_ssl_client_policy
- oracle/wss_saml20_token_bearer_over_ssl_client_policy
- oracle/wss_saml20_token_over_ssl_client_policy
- oracle/wss_username_token_client_policy
- oracle/wss_username_token_over_ssl_client_policy
- oracle/wss10_message_protection_client_policy
- oracle/wss10_saml_token_with_message_integrity_client_policy

- oracle/wss10_saml_token_with_message_protection_client_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy
- oracle/wss10_saml20_token_with_message_protection_client_policy
- oracle/wss10_username_id_propagation_with_msg_protection_client_policy
- oracle/wss10_username_token_with_message_protection_client_policy
- oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy
- oracle/wss10_x509_token_with_message_protection_client_policy
- oracle/wss11_message_protection_client_policy
- oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy
- oracle/wss11_saml_token_with_message_protection_client_policy
- oracle/wss11_saml20_token_with_message_protection_client_policy
- oracle/wss11_username_token_with_message_protection_client_policy
- oracle/wss11_x509_token_with_message_protection_client_policy

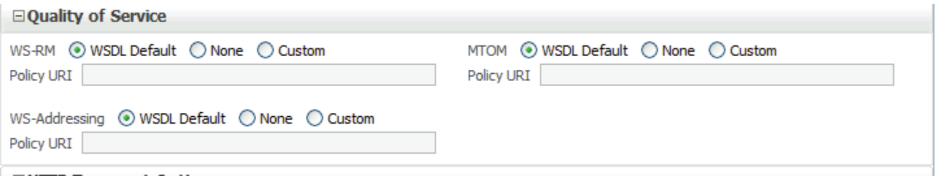
12.4 Enabling Quality of Service Testing

Note: This section is not applicable when testing RESTful Web services.

Three characteristics of Quality of Service (QoS) can be tested: reliable messaging (WS-RM), WS-Addressing, and Message Transmission Optimization Mechanism (MTOM) in the Quality of Service section of the Test Web Service Page (Figure 12–7). For each type of Quality of Service, there are three options:

- **WSDL Default** – Execute the default behavior of the WSDL. For example, if **WSDL Default** is selected for MTOM, and the WSDL contains a reference to an MTOM policy, the policy is enforced. If the WSDL does not contain a reference to an MTOM policy, then no MTOM policy is enforced.
- **None** – No policy for the specific QoS, even if it is included in the WSDL, is executed. For example, if **None** is selected for WS-RM, no reliable messaging policy is enforced. If the WSDL contains a reference to a reliable messaging policy, it is ignored.
- **Custom** – Enforce a custom policy. For example, if a WS-Addressing policy is referenced in the WSDL, this policy will be ignored, and the policy specified in **URI** will be used instead.
- **URI** – Specify the location of the policy to be enforced.

Figure 12–7 Quality of Service Parameters on the Test Web Service Page



⊟ Quality of Service

WS-RM ● WSDL Default ○ None ○ Custom

Policy URI

MTOM ● WSDL Default ○ None ○ Custom

Policy URI

WS-Addressing ● WSDL Default ○ None ○ Custom

Policy URI

12.5 Enabling HTTP Transport Options

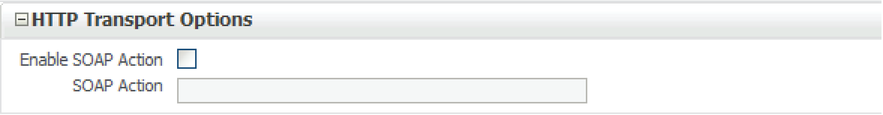
> **Note:** This section is not applicable when testing RESTful Web services.

The test mechanism uses the WSDL to determine whether a SOAP action is available to test. If the WSDL soap:operation has a soapAction attribute, then this is displayed and **Enable SOAP Action** is enabled.

When a request is sent with **Enable SOAP Action** enabled, then the SOAP action HTTP header is sent.

To change this behavior, clear the **Enable SOAP Action** box, in which case the HTTP header is not sent. Or, you can override the behavior by providing a different value in the **SOAP Action** field. (You must already know the SOAP action that you want to test, and the syntax.)

Figure 12–8 HTTP Transport Options on the Test Web Service Page



⊟ HTTP Transport Options

Enable SOAP Action □

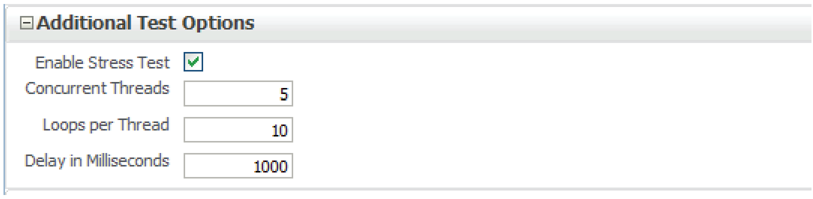
SOAP Action

12.6 Stress Testing the Web Service Operation

Select the **Enable Stress Test** check box (Figure 12–9) to invoke a continuous series of invocations of the Web service operation (Figure 12–9). The following options are available:

- **Concurrent Threads** – The number of concurrent threads on which the invocations should be sent. The default is 5 threads.
- **Loops per Thread** – The number of times to invoke the operation. The default is 10 times.
- **Delay in Milliseconds** – The number of milliseconds to wait between operation invocations. The default is 1000 milliseconds (1 second).

Figure 12–9 Stress Testing Parameters on the Test Web Service Page



⊟ Additional Test Options

Enable Stress Test ☑

Concurrent Threads 5

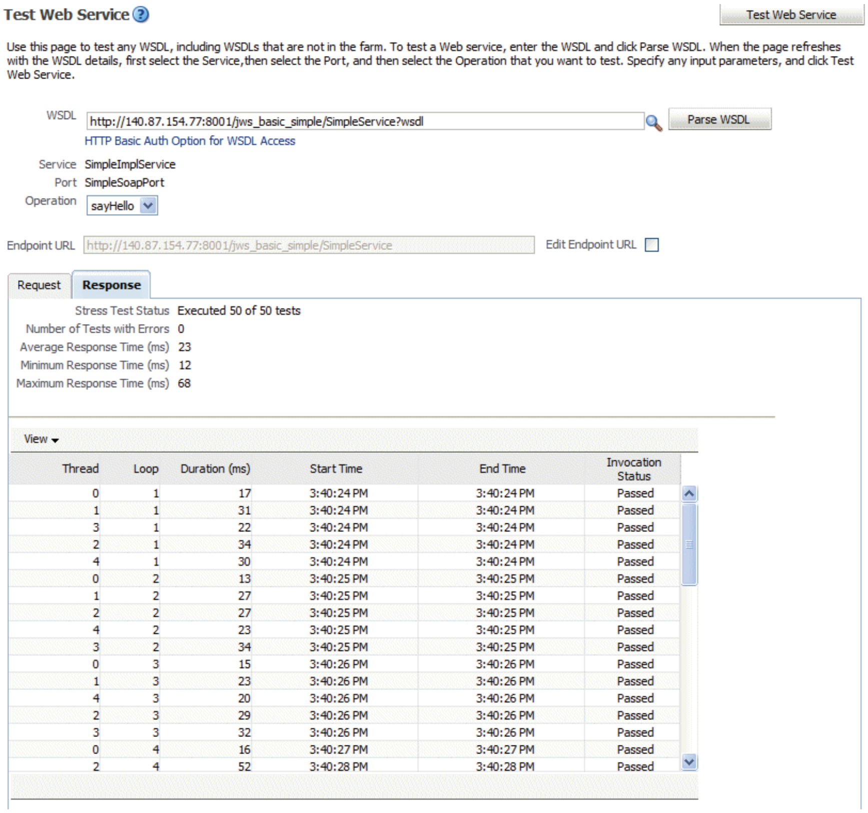
Loops per Thread 10

Delay in Milliseconds 1000

When you invoke the test, a progress box indicates the test status. When the stress test is complete, a confirmation page displays the results of the test.

The **Response** tab provides additional information about the stress test, including the number of tests with errors, and the average, minimum, and maximum response times. Details about each test are provided in tabular form. For each test, you can view the thread and loop numbers, the duration of the test, the start and end times for the test, and the invocation status. You can filter the fields displayed in the table using the **View** menu.

Figure 12–10 Stress Test Results on Test Web Service Page



Test Web Service

Use this page to test any WSDL, including WSDLs that are not in the farm. To test a Web service, enter the WSDL and click Parse WSDL. When the page refreshes with the WSDL details, first select the Service,then select the Port, and then select the Operation that you want to test. Specify any input parameters, and click Test Web Service.

WSDL http://140.87.154.77:8001/jws_basic_simple/SimpleService?wsdl

HTTP Basic Auth Option for WSDL Access

Service SimpleImplService

Port SimpleSoapPort

Operation sayHello

Endpoint URL http://140.87.154.77:8001/jws_basic_simple/SimpleService

Edit Endpoint URL □

Stress Test Status Executed 50 of 50 tests

Number of Tests with Errors 0

Average Response Time (ms) 23

Minimum Response Time (ms) 12

Maximum Response Time (ms) 68

Thread	Loop	Duration (ms)	Start Time	End Time	Invocation Status
0	1	17	3:40:24 PM	3:40:24 PM	Passed
1	1	31	3:40:24 PM	3:40:24 PM	Passed
3	1	22	3:40:24 PM	3:40:24 PM	Passed
2	1	34	3:40:24 PM	3:40:24 PM	Passed
4	1	30	3:40:24 PM	3:40:24 PM	Passed
0	2	13	3:40:25 PM	3:40:25 PM	Passed
1	2	27	3:40:25 PM	3:40:25 PM	Passed
2	2	27	3:40:25 PM	3:40:25 PM	Passed
4	2	23	3:40:25 PM	3:40:25 PM	Passed
3	2	34	3:40:25 PM	3:40:25 PM	Passed
0	3	15	3:40:26 PM	3:40:26 PM	Passed
1	3	23	3:40:26 PM	3:40:26 PM	Passed
4	3	20	3:40:26 PM	3:40:26 PM	Passed
2	3	29	3:40:26 PM	3:40:26 PM	Passed
3	3	32	3:40:26 PM	3:40:26 PM	Passed
0	4	16	3:40:27 PM	3:40:27 PM	Passed
2	4	52	3:40:28 PM	3:40:28 PM	Passed

12.7 Disabling the Test Page for a Web Service

Note: This section does not apply to Java EE Web services.

Disabling the test page for a Web service allows you to increase security by reducing the externally visible details of an application that exposes Web services.

Note: Disabling the **Endpoint Test Enabled** control affects only the Web service's externally-visible test page. It does not affect the Web service test feature described in this chapter.

To disable the Test Page using Fusion Middleware Control

1. Navigate to the Web Services summary page, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.

2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of the port to navigate to the Web Service Endpoint page.
4. Click the **Configuration** tab.
5. In the **Endpoint Test Enabled** field, select **False** from the list.
6. Click **Apply**.

13

Monitoring the Performance of Web Services

This chapter describes how to monitor the performance of a Web service using Fusion Middleware Control. The chapter includes the following sections:

- Overview of Performance Monitoring
- When Are Web Service Statistics Started or Reset?
- Viewing Web Service Statistics for an Application
- Viewing Web Service Statistics for a Server Instance
- Viewing Web Service Statistics for an Individual Web Service
- Viewing Operation Statistics for a Web Service Endpoint
- Viewing Web Service Statistics for Java EE Web Service Clients
- Viewing the Security Violations for a Web Service

In addition to the monitoring features described in this chapter, see "Analyzing Policy Usage" on page 7-26 to analyze how policies are used by one or more Web services.

13.1 Overview of Performance Monitoring

Note: Not all of the monitoring features described in this chapter apply to Java EE Web services.

From the Web Services home page in Fusion Middleware Control, you can do the following:

- Monitor Web services faults, including Security, Reliable Messaging, MTOM, Management, and Service faults.
- Monitor Security failures, including authentication, authorization, message integrity, and message confidentiality failures.
- Configure your Web services ports, including enabling and disabling the port, attaching policies to Web services, and enabling or disabling policies.

The Application home page also displays select Web service details if the application includes Web services.

13.2 When Are Web Service Statistics Started or Reset?

The statistics described in this chapter are started or reset when any one of the following events occur:

- When the application is being deployed for the first time.
- When the application is redeployed.
- If the application is already deployed, and the hosting server is restarted.

13.3 Viewing Web Service Statistics for an Application

In Fusion Middleware Control, the Web Services summary page for an application displays the collective **Summary** and fault/violation information for all Web services in the application, as shown in Figure 13–1.

The **Charts** section shows a graphical view of all security faults for a Web service.

To navigate to the Web Service Summary page for a Web service

1. From the navigator pane, click the plus sign (+) for the Application Deployments folder to expose the applications in the domain, and select the application.

   The Application Deployment home page is displayed.

2. Using Fusion Middleware Control, click **Application Deployment**, then click **Web Services**.

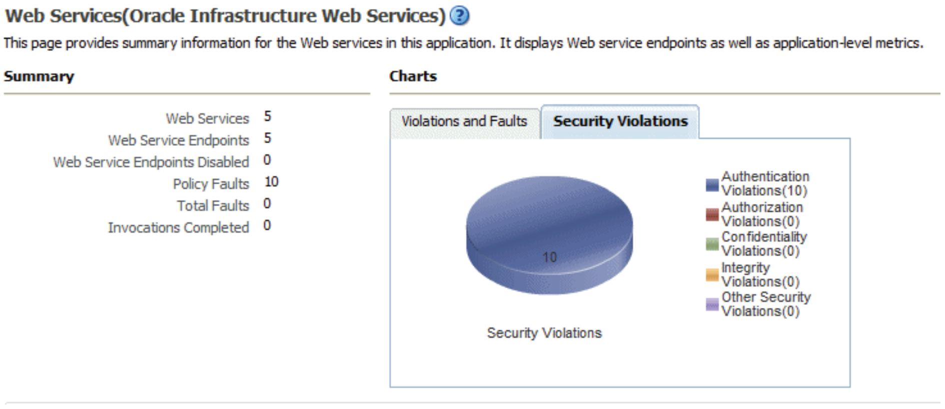
   The **Web Services Summary** page for this application is displayed.

   The page displays Web service endpoints as well as application-level metrics.

For Oracle Infrastructure Web services, the following Web service-wide statistics are displayed:

- Web Services (Number of Web services in the application)
- Web Service Endpoints
- Web Service Endpoints Disabled
- Total Policy Violations
- Total Faults
- Invocations Completed

Figure 13–1 Web Services Performance Summary and Charts



Web Services(Oracle Infrastructure Web Services)

This page provides summary information for the Web services in this application. It displays Web service endpoints as well as application-level metrics.

Summary

Web Services	5
Web Service Endpoints	5
Web Service Endpoints Disabled	0
Policy Faults	10
Total Faults	0
Invocations Completed	0

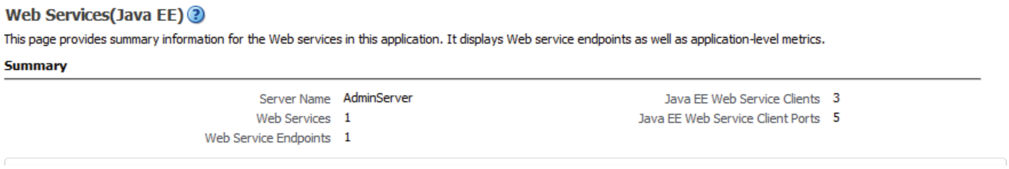
Charts
Violations and Faults
Security Violations
10
Security Violations
Authentication Violations(10)
Authorization Violations(0)
Confidentiality Violations(0)
Integrity Violations(0)
Other Security Violations(0)

For WebLogic Java EE Web services, the following Web service-wide statistics are displayed:

- Server Name (server on which the application is running)

- Web Services (number of Web services in the application)
- Web Service Endpoints
- Java EE Web Service Clients (number of run-time client instances in the application)
- Java EE Web Service Client Ports (number of Web service client ports in the application to which you can attach Oracle WSM policies)

Figure 13–2 Java EE Web Services Summary



Web Services(Java EE) ?

This page provides summary information for the Web services in this application. It displays Web service endpoints as well as application-level metrics.

Summary

Server Name	AdminServer	Java EE Web Service Clients	3
Web Services	1	Java EE Web Service Client Ports	5
Web Service Endpoints	1		

13.4 Viewing Web Service Statistics for a Server Instance

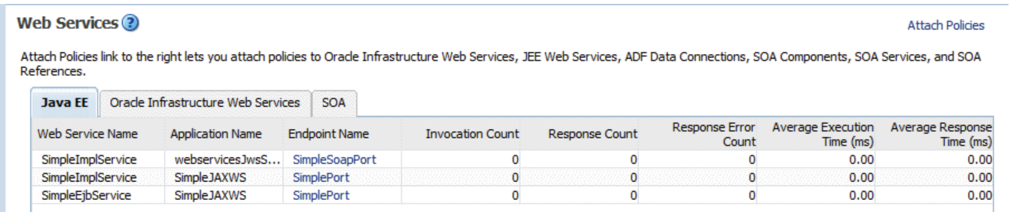
The server-side Web services page displays statistics for all of the Web services on that server.

To view the Web service statistics for a server

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you want to see the policies and select the domain.
2. Expand the domain to show the servers in that domain. Select the server for which you want to view the statistics.
3. Using Fusion Middleware Control, click **WebLogic Server**, and then **Web Services**.
4. The Web services statistics page for the server is displayed, as shown in Figure 13–3.

   Depending on the types of Web services you have deployed, tabs are available for the following Web service types: Java EE, Oracle Infrastructure Web Services, and SOA.

Figure 13–3 Web Services for a Server



Web Services ?

Attach Policies

Attach Policies link to the right lets you attach policies to Oracle Infrastructure Web Services, JEE Web Services, ADF Data Connections, SOA Components, SOA Services, and SOA References.

Web Service Name	Application Name	Endpoint Name	Invocation Count	Response Count	Response Error Count	Average Execution Time (ms)	Average Response Time (ms)
SimpleImplService	webservicesJwsS...	SimpleSoapPort	0	0	0	0.00	0.00
SimpleImplService	SimpleJAXWS	SimplePort	0	0	0	0.00	0.00
SimpleEjbService	SimpleJAXWS	SimplePort	0	0	0	0.00	0.00

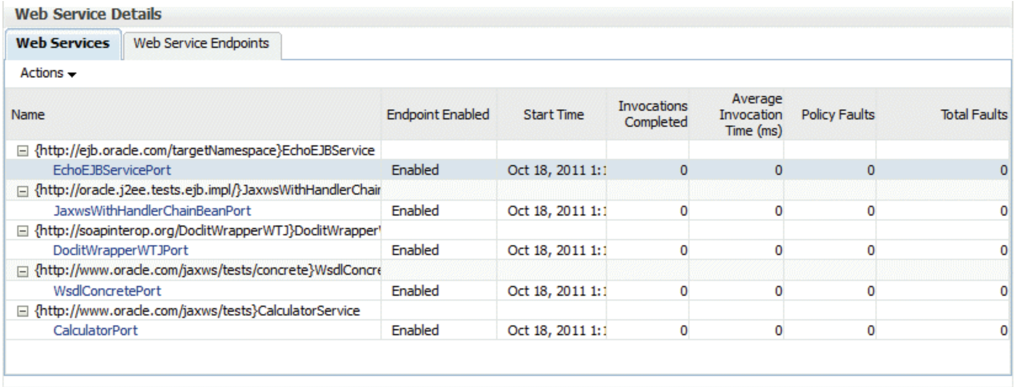
13.5 Viewing Web Service Statistics for an Individual Web Service

The **Web Service Details** section of the Web Services Summary page displays statistics on a per-Web service basis, as shown in Figure 13–4. The following statistics are displayed:

- Name of the Web service. Click the plus sign (+) for the Web service to display the Web service endpoint.

- Endpoint Enabled—Flag that specifies whether the Web service is enabled or disabled. For Oracle Infrastructure Web service providers, this field displays n/a.
- Start Time—Time the Web service was started.
- Invocations Completed—Number of completed requests to this endpoint.
- Average Invocation Time—Average time for all Web service invocations to be processed.
- Policy Faults—Number of failed requests because a policy was not successfully executed.
- Total Faults—Total number of failed requests.

Figure 13–4 Web Service-Specific Statistics



Name	Endpoint Enabled	Start Time	Invocations Completed	Average Invocation Time (ms)	Policy Faults	Total Faults
⊟ {http://ejb.oracle.com/targetNamespace}EchoEJBService						
EchoEJBServicePort	Enabled	Oct 18, 2011 1:1	0	0	0	0
⊟ {http://oracle.j2ee.tests.ejb.impl/}JaxwsWithHandlerChair						
JaxwsWithHandlerChainBeanPort	Enabled	Oct 18, 2011 1:1	0	0	0	0
⊟ {http://soapinterop.org/DoclitWrapperWTJ}DoclitWrapper						
DoclitWrapperWTJPort	Enabled	Oct 18, 2011 1:1	0	0	0	0
⊟ {http://www.oracle.com/jaxws/tests/concrete}WsdlConcre						
WsdlConcretePort	Enabled	Oct 18, 2011 1:1	0	0	0	0
⊟ {http://www.oracle.com/jaxws/tests}CalculatorService						
CalculatorPort	Enabled	Oct 18, 2011 1:1	0	0	0	0

13.6 Viewing Operation Statistics for a Web Service Endpoint

To display operation statistics for a particular Web service endpoint:

1. In the Web Services Details section of the Web Services summary page, select the **Web Service Endpoints** tab.
2. Select the endpoint for which you want to display the statistics.

   The Web Service Endpoint page is displayed.
3. Select the **Operations** tab if it is not already selected.

   The following statistics are presented for Oracle Infrastructure Web services:

   - Operation Name—Name of the operation.
   - One Way—Flag that specifies whether the operation returns a value to the calling operation.
   - Action—URI of the action.
   - Input Encoding—Encoding style of the input message.
   - Output Encoding—Encoding style of the output message.
   - Invocations Completed—Number of completed requests to this endpoint.
   - Average Invocation Time—Average time for all Web service invocations to be processed.
   - Faults—Total number of faults for this endpoint.

   The following statistics are presented for WebLogic Java EE Web services:

- Name—Name of the operation.
- Invocation Count—Number of times that the Web service was invoked.
- Dispatch Time Average—Average time for all Web service invocations to be processed.
- Execution Time Average—Average time for all Web service executions.
- Response Time Average—Average time for all responses generated.
- Response Count—Total number of responses generated from the Web service invocations.
- Response Error Count—Total number of errors from responses generated from the Web service invocations.

13.7 Viewing Web Service Statistics for Java EE Web Service Clients

To display Web Service statistics for the run-time client instances in a Java EE application:

1. Navigate to the home page for the Java EE Web service, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. Select the **Java EE Web Service Clients** tab to view the clients in the application.

   Note: This tab is available only if the application contains Java EE Web service clients.

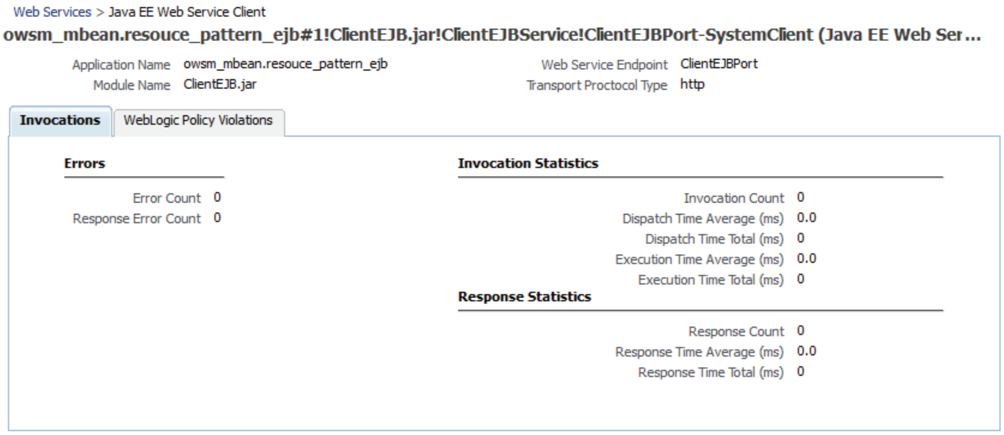
3. Select the **Monitoring** tab, if it is not already selected to view the statistics for all run-time client instances in the application.

   Note: For JAX-WS Web services, the Web services run time creates system-defined client instances within a Web service endpoint that are used to send protocol-specific messages as required by that endpoint. These client instances are named after the Web service endpoint that they serve with the following suffix: -SystemClient. Monitoring information relevant to the system-defined client instances is provided to assist in evaluating the application.

4. Select the client in the Client column to display Web service statistics for that client.

   The Java EE Web Service Client page is displayed, as shown in Figure 13–5.

Figure 13–5 Java EE Web Service Client Statistics


Web Services > Java EE Web Service Client
Invocations
WebLogic Policy Violations

owsm_mbean.resouce_pattern_ejb#1!ClientEJB.jar!ClientEJBService!ClientEJBPort-SystemClient (Java EE Web Ser...

Application Name owsm_mbean.resouce_pattern_ejb
Module Name ClientEJB.jar
Web Service Endpoint ClientEJBPort
Transport Proctocol Type http

Errors

Error Count 0
Response Error Count 0

Invocation Statistics

Invocation Count 0
Dispatch Time Average (ms) 0.0
Dispatch Time Total (ms) 0
Execution Time Average (ms) 0.0
Execution Time Total (ms) 0

Response Statistics

Response Count 0
Response Time Average (ms) 0.0
Response Time Total (ms) 0

The following summary information is presented for the run-time client instance.

- Application Name—The name of the application with which the client is associated.
- Module Name—Name of the Java EE module in which the endpoint is running.
- Web Service Endpoint—Name of the port which the client invokes.
- Transport Protocol Type—Transport protocol required by the service.

5. Select the **Invocations** tab to view the invocation statistics for the client.

   Table 13–1 lists the invocation statistics displayed for the run-time client instance.

Table 13–1 Invocation Statistics for Java EE Web Service Client

Element	Description
Errors	
Error Count	Total number of security faults and violations.
Response Error Count	Total number of errors from responses generated from invocations of this client instance
Invocation Statistics	
Invocation Count	Total number of times that operations on service side have been invoked by the client instance in the current measurement period.
Dispatch Time Average (ms)	Average dispatch time for the current measurement period.
Dispatch Time Total (ms)	Total time for all dispatches of this operation in the current measurement period.
Execution Time Average (ms)	Average execution time of this operation.
Execution Time Total (ms)	Total time for all executions of this operation
Response Statistics	
Response Count	Total number of responses generated from invocations of this operation.
Response Time Average (ms)	Average response time from the responses generated from invocations of this operation.

Table 13–1 (Cont.) Invocation Statistics for Java EE Web Service Client

Element	Description
Response Time Total (ms)	Total time for all responses generated from invocations of this operation.

6. Select the **WebLogic Policy Violations** tab to the statistics for policy violations of this client run-time instance.

   lists the policy violations for the client run-time instance.

Table 13–2 WebLogic Policy Violations for Java EE Web Service Client

Element	Description
Summary	
Total Faults	Total number of failed requests.
Policy Faults	Total number of policy faults.
Total Security Faults	Total number of security faults and violations.
Violations	
Authentication Violations	Total number of authentication violations generated for this port. Only incoming message processing can add to the violation count.
Confidentiality Violations	Total number of confidentiality violations generated for this port. Both outgoing and incoming message processing can add to the violation count.
Integrity Violations	Total number of integrity violations generated for this port. Both outgoing and incoming message processing can add to the violation count.
Successes	
Authentication Successes	Total number of authentication successes detected for this port. Only incoming message processing can add to the success count.
Confidentiality Successes	Total number of confidentiality successes generated for this port. Both outgoing and incoming message processing can add to the success count.
Integrity Successes	Total number of integrity successes generated for this port. Both outgoing and incoming message processing can add to the success count.

13.8 Viewing the Security Violations for a Web Service

Follow the procedure below to view security violations for a Web service.

To view the security violations for an Oracle Infrastructure Web service:

1. Navigate to the Web Services Summary page as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. In the Charts section of the page, select the **Security Violations** tab.

   A graphical representation of the authentication, authorization, confidentiality, and integrity faults for all Web services in the application is displayed in the pie chart.
3. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service endpoints if they are not already displayed.

4. Click the name of the endpoint to navigate to the Web Service Endpoint page.
5. Click the **Charts** tab to see a graphical representation of all faults and all security violations for the endpoint.
6. Click the **OWSM Policies** tab.

   Two tables are displayed.

   The Globally Attached Policies table displays the name of the policy and the policy set that references it.

   The Directly Attached Policies table displays the name of the policy and the policy status (whether the policy is enabled or disabled).

   Both tables list the category to which the policy belongs (security, MTOM attachments, reliable messaging, WS-addressing, and management).

   Table 13–3 lists the violation information provided for each type of policy attachment.

Table 13–3 Policy Violation Information for an Endpoint

Violation Type	Description
Total Violations	Total number of faults for this policy. **Note**: Total violations may not be equal to the sum of the security violations shown below (for example, Authentication, Authorization, Confidentiality, and Integrity). Other security violations that do not fall into these major categories and non-security violations are also captured in the total violations count.
Security Violations	
Authentication	Number of authentication failures since the server was restarted.
Authorization	Number of authorization failures since the server was restarted.
Confidentiality	Number of message confidentiality failures since the server was restarted.
Integrity	Number of message integrity failures since the server was restarted.

To view the security violations for a WebLogic JAX-WS Web service:

1. Navigate to the Web Services Summary page as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service endpoints if they are not already displayed.
3. Click the name of the endpoint to navigate to the Web Service Endpoint page.
4. Do one of the following, depending on the type of policies attached to the endpoint:
   - If Oracle WSM policies are attached to the endpoint, click the **OWSM Policies** tab.

     A list of the policies that are attached to the endpoint is displayed. For each policy, the table displays the name of the policy, the category of the policy (security, MTOM attachments, reliable messaging, WS-addressing, and management), and the policy status (whether the policy is enabled or disabled). Table 13–3 describes the violation information that is displayed for each Oracle WSM policy attached to the endpoint.

- If WebLogic policies are attached to the endpoint, click the **WebLogic Policy Violations** tab.

  This tab shows policy violation details about WebLogic policies attached to a JAX-WS endpoint. Table 13–4 describes the information provided on this page.

Table 13–4 WebLogic Policy Violation Data

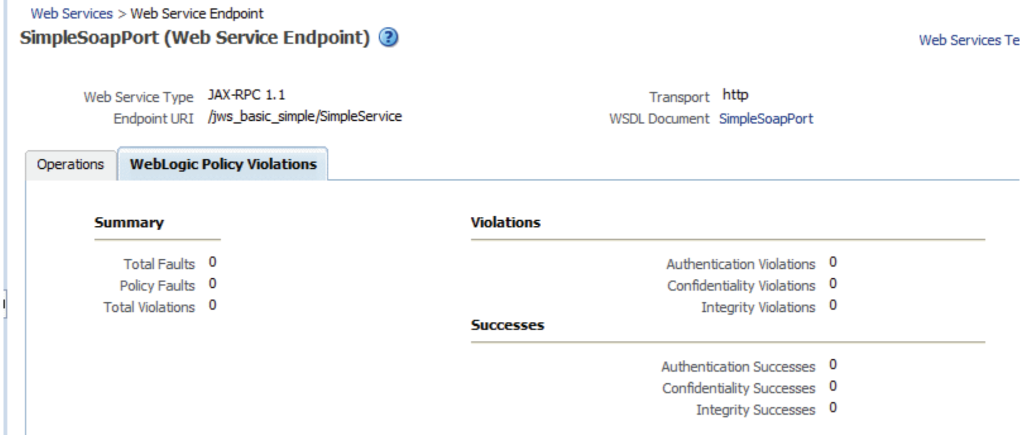
Element	Description
Summary	
Total Faults	Total number of failed requests.
Policy Faults	Number of failed requests because a policy was not successfully executed.
Total Violations	Total number of faults for this policy.
Violations	
Authentication Violations	Number of authentication failures since the server was restarted.
Confidentiality Violations	Number of message confidentiality failures since the server was restarted.
Integrity Violations	Number of message integrity failures since the server was restarted.
Successes	
Authentication Successes	Number of authentication successes since the server was restarted.
Confidentiality Successes	Number of message confidentiality successes since the server was restarted.
Integrity Successes	Number of message integrity successes since the server was restarted.

To view the security violations for a WebLogic JAX-RPC Web service:

1. Navigate to the Web Services Summary page for the application.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service endpoints if they are not already displayed.
3. Click the name of the endpoint to navigate to the Web Service Endpoint page.
4. Click the **WebLogic Policy Violations** tab.

   This tab shows policy violation details about WebLogic policies attached to a JAX-RPC endpoint, as shown in Figure 13–6. For a description of the information displayed on this tab, see Table 13–4.

Figure 13–6 Security Violations for a WebLogic JAX-RPC Web Service Endpoint



Web Services > Web Service Endpoint

SimpleSoapPort (Web Service Endpoint)

Web Services Te

Web Service Type	JAX-RPC 1.1	Transport	http
Endpoint URI	/jws_basic_simple/SimpleService	WSDL Document	SimpleSoapPort

Summary

Total Faults	0
Policy Faults	0
Total Violations	0

Violations

Authentication Violations	0
Confidentiality Violations	0
Integrity Violations	0

Successes

Authentication Successes	0
Confidentiality Successes	0
Integrity Successes	0

Part III

Advanced Administration

Part III contains the following chapters:

14 Advanced Administration

This chapter includes the following sections:

14.1 Registering Web Services and Sources

A key feature of the Web services model is the ability to make Web services widely available and discoverable. UDDI is one approach to publishing and discovery of Web services that centralizes information about businesses and their services in registries. Another emerging alternative standard is the Web Services Inspection Language (WSIL) specification.

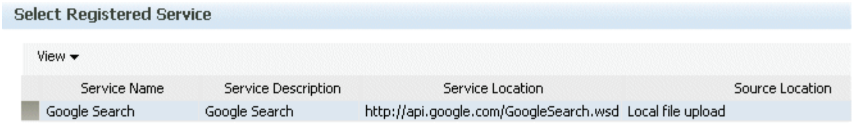
Oracle Enterprise Manager Fusion Middleware Control provides support for registering Web services that are published in WSIL documents and UDDI v3 registries. Any service that is available in a WSIL document or a UDDI v3 registry can be registered within Enterprise Manager.

You can also register meta information, or a profile, for sources of services to help you manage your registered services within Enterprise Manager. Once you register a source and assign it a logical name, you do not need to specify connectivity information, such as a URL for a WSDL, in the future. A domain can have multiple registered sources, and each registered source can have multiple registered services. Once you register a source, you can easily look up services that you can register to the source.

Service names and corresponding WSDLs must be unique within a registered single source. Once you have registered a service, an attempt to register another service with the same name, or a different name but the same WSDL URL as another service, is not valid.

Once you register a Web service, you can later, more conveniently, reference the service from a selection list within Enterprise Manager. For example, when testing a Web service as described in "Testing Your Web Services" on page 12-1, instead of specifying a WSDL, you can click the **Search** icon and then select the WSDL from the list of registered services, as shown in Figure 14–1.

Figure 14–1 Selecting From a Registered Service



Select Registered Service

Service Name	Service Description	Service Location	Source Location
Google Search	Google Search	http://api.google.com/GoogleSearch.wsd	Local file upload

14.1.1 UDDI Basics

Universal Description Discovery & Integration (UDDI) is an industry initiative that aims to enable businesses to quickly, easily, and dynamically find and carry out transactions with one another. A populated UDDI registry contains cataloged information about businesses; the services that they offer; and communication standards and interfaces they use to conduct transactions.

The owners of Web services publish them to the UDDI registry. Once published, the UDDI registry maintains pointers to the Web Service description and to the service. The UDDI allows clients to search this registry, find the intended service, and retrieve its details. These details include the service invocation point as well as other information to help identify the service and its functionality.

14.1.2 WSIL Basics

WSIL defines an Extensible Markup Language (XML) format for referencing Web service descriptions. These references are contained in a WSIL document, and refer to Web service descriptions (for example, WSDL files) and to other aggregations of Web services (for example, another WSIL document or a UDDI registry).

WSIL documents are typically distributed by the Web service provider. These documents describe how to inspect the provider's Web site for available Web services. Therefore, the WSIL standard also defines rules for how WSIL documents should be made available to consumers of Web services.

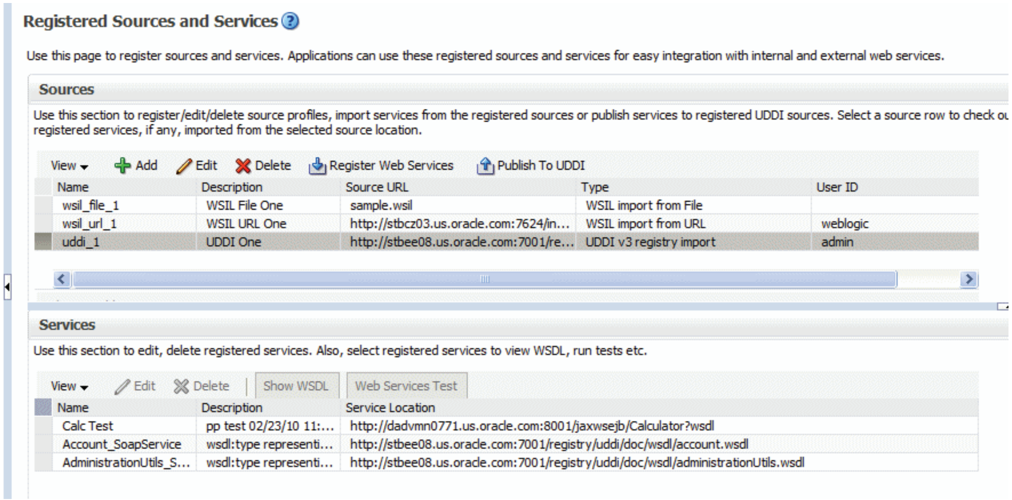
The WSIL model decentralizes Web service discovery. In contrast to UDDI registries, which centralize information on multiple business entities and services, WSIL makes it possible to provide Web service description information from any location. Unlike UDDI, WSIL is not concerned about business entity information, and does not require a specific service description format. It assumes that you know who the service provider is and relies on other standards for Web service description, such as WSDL.

14.1.3 Viewing Registered Sources and Web Services

Follow the steps in this section to view and edit a registered source and Web service.

1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to view the registered sources and Web services.
2. Select the domain.
3. Using Fusion Middleware Control, select **WebLogic Domain** then **Web Services** and then **Registered Services**. The Registered Sources and Services page appears, as shown in Figure 14–2.

Figure 14–2 Viewing Registered Sources and Services



Registered Sources and Services

Use this page to register sources and services. Applications can use these registered sources and services for easy integration with internal and external web services.

Sources

Use this section to register/edit/delete source profiles, import services from the registered sources or publish services to registered UDDI sources. Select a source row to check ou registered services, if any, imported from the selected source location.

Name	Description	Source URL	Type	User ID
wsil_file_1	WSIL File One	sample.wsil	WSIL import from File	
wsil_url_1	WSIL URL One	http://stbcz03.us.oracle.com:7624/in...	WSIL import from URL	weblogic
uddi_1	UDDI One	http://stbee08.us.oracle.com:7001/re...	UDDI v3 registry import	admin

Services

Use this section to edit, delete registered services. Also, select registered services to view WSDL, run tests etc.

Name	Description	Service Location
Calc Test	pp test 02/23/10 11:...	http://dadvmn0771.us.oracle.com:8001/jaxwsejb/Calculator?wsdl
Account_SoapService	wsdl:type representi...	http://stbee08.us.oracle.com:7001/registry/uddi/doc/wsdl/account.wsdl
AdministrationUtils_S...	wsdl:type representi...	http://stbee08.us.oracle.com:7001/registry/uddi/doc/wsdl/administrationUtils.wsdl

In the Sources table, you can view the following information about each registered source:

- Name—Logical name for the source
- Description—Description of the source
- Source URL—Location of the source in URL format
- Type—Source type: UDDI v3 registry import, WSIL import from file, WSIL import from URL
- User ID—User ID for the external source

You can customize the information that is displayed using the **View** menu. From this section of the page, you can also add new sources, edit or delete sources, register Web services for a source, and publish a Web service from a source to a predefined UDDI registry.

4. Select a source in the Sources table.

Each registered source can have multiple registered services. In the Services table, you can view the following information about the registered services imported from the selected source location:

- Name—Name of the registered service
- Description—Description of the service
- Service location—Location of the service in URL format

You can customize the information that is displayed using the **View** menu. You can also display the WSDL for the selected service and test the selected Web service.

14.1.4 Registering a Source

You can register Web service sources of the following types:

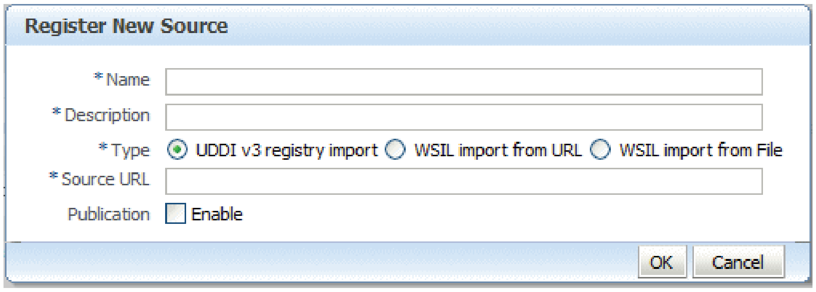
- UDDI v3 registry import
- WSIL import from URL
- WSIL import from file

Follow the steps in this section to register a source.

To register a source

1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to register a Web service source.
2. Select the domain.
3. Using Fusion Middleware Control, select **WebLogic Domain** then **Web Services** and then **Registered Services**. The Registered Sources and Services page appears, as shown in Figure 14–2.
4. Click **Add** to register a new source. The Register New Source page appears, as shown in Figure 14–3.

Figure 14–3 Register New Source Page



Register New Source

* Name

* Description

* Type ● UDDI v3 registry import ○ WSIL import from URL ○ WSIL import from File

* Source URL

Publication □ Enable

5. Enter the following information for the new source.
   - **Name**—A logical name for the source.
   - **Description**—A description of the source.
   - **Type**—choose from one of three options: **UDDI v3 registry import**, **WSIL import from URL**, or **WSIL import from File**

     Additional information that you must enter differs based on the option you select.
6. If you selected **UDDI v3 registry import**, enter the following information:
   - In the **Source URL** field, enter the UDDI inquiry URL, for example, `http://somehost/uddi/inquiry`.
   - To allow the services to be published to a UDDI source (which is an external UDDI registry), select the **Enable** box and complete the fields as follows:

- In the **Publication URL** field, enter URL location of the registry to which you want to publish the service.
- In the **Security URL** field, enter the URL location of the security port required to access the registry.
- In the **User ID** and **Password** fields, enter the security credentials required to access the registry.

7. If you selected **WSIL import from URL**, enter the following information:
   - In the **Source URL** field, enter the location of the WSIL in URL form.
   - If a username and password are required to access the WSIL, select the **Enable** box in the **Basic Authorization** field. In the **User ID** and **Password** fields, enter the username and password.
8. If you selected **WSIL import from File**, click **Browse** (next to the **WSIL File** field) to select the WSIL file to be imported.
9. Click **OK** to register the source.

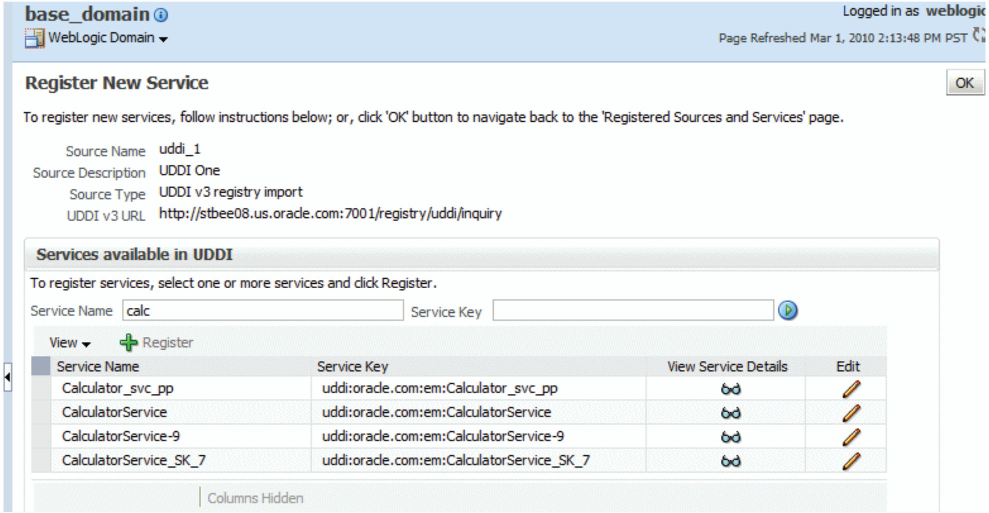
14.1.5 Registering Web Services from a UDDI Source

Follow the steps in this section to register Web services from a registered UDDI source.

1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to register a Web service.
2. Select the domain.
3. Using Fusion Middleware Control, select **WebLogic Domain** then **Web Services** and then **Registered Services**. The Registered Sources page displays, as shown in Figure 14–1.
4. Select the UDDI source from which you want to register services. Note that the Type for a UDDI source is specified as UDDI v3 registry import.
5. Select **Register Web Services**.

   The Register New Service page displays, as shown in Figure 14–4.

Figure 14–4 Register New Service from UDDI Source



Register New Service

To register new services, follow instructions below; or, click 'OK' button to navigate back to the 'Registered Sources and Services' page.

Source Name uddi_1
Source Description UDDI One
Source Type UDDI v3 registry import
UDDI v3 URL http://stbee08.us.oracle.com:7001/registry/uddi/inquiry

Services available in UDDI

To register services, select one or more services and click Register.

Service Name calc
Service Key

Service Name	Service Key	View Service Details	Edit
Calculator_svc_pp	uddi:oracle.com:em:Calculator_svc_pp	👓	✎
CalculatorService	uddi:oracle.com:em:CalculatorService	👓	✎
CalculatorService-9	uddi:oracle.com:em:CalculatorService-9	👓	✎
CalculatorService_SK_7	uddi:oracle.com:em:CalculatorService_SK_7	👓	✎

The Register New Service page displays the source information, in read-only format, and a list of the services that are available in UDDI that you can register.

You can filter the list of available services that are displayed using the **Service Name** and **Service Key** fields. For example, to find calculator services, enter `calc` in the **Service Name** field. Only services that contain the `calc` string, such as calculator services are displayed. The search is not case-sensitive.

In the Services available in UDDI section of the page, you can view service details from UDDI for each service in the table by clicking the **View Service Details** icon. The Service Details from UDDI window displays information about the service such as the Service Name, Service Description, Service WSDL, Service Key, Business Key, and Service Location, among others.

You can edit the details of a service by clicking the **Edit** icon, which allows you to change the name and description of the selected service.

6. In the Services available in UDDI section of the page, select the service or services that you want to register from the source and click **Register**.

   A confirmation message displays indicating that the service was registered successfully.

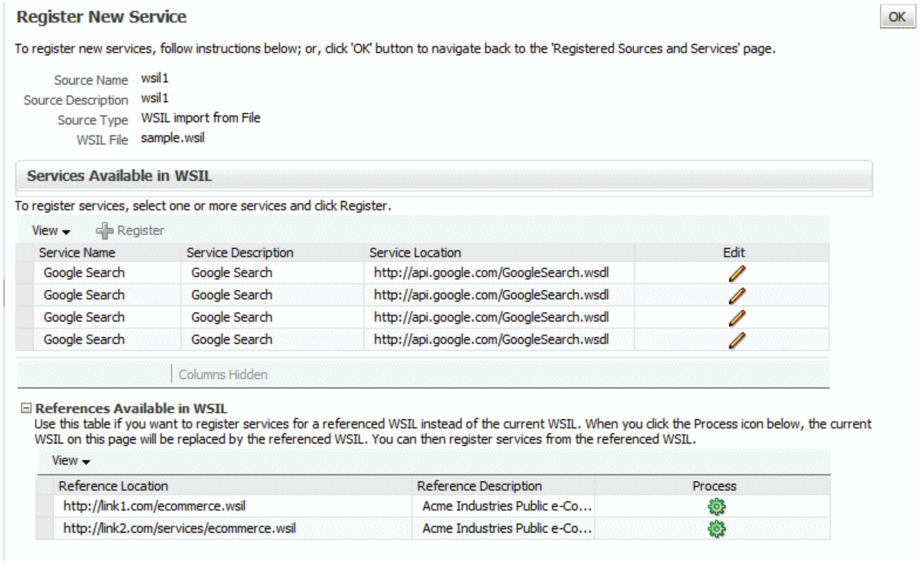
14.1.6 Registering Web Services from a WSIL Source

Follow the steps in this section to register Web services from a registered WSIL source.

1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to register a Web service.
2. Select the domain.
3. Using Fusion Middleware Control, select **WebLogic Domain** then **Web Services** and then **Registered Services**. The Registered Sources and Services page displays, as shown in Figure 14–1.
4. Select the WSIL source from which you want to register services. Note that the Type for a WSIL source is specified as either WSIL import from File or WSIL import from URL.
5. Select **Register Web Services**.

   The Register New Service page displays, as shown in Figure 14–5.

Figure 14–5 Register New Service from WSIL Source



Register New Service

To register new services, follow instructions below; or, click 'OK' button to navigate back to the 'Registered Sources and Services' page.

Source Name	wsil1
Source Description	wsil1
Source Type	WSIL import from File
WSIL File	sample.wsil

Services Available in WSIL

To register services, select one or more services and click Register.

Service Name	Service Description	Service Location	Edit
Google Search	Google Search	http://api.google.com/GoogleSearch.wsdl	✎
Google Search	Google Search	http://api.google.com/GoogleSearch.wsdl	✎
Google Search	Google Search	http://api.google.com/GoogleSearch.wsdl	✎
Google Search	Google Search	http://api.google.com/GoogleSearch.wsdl	✎

⊟ **References Available in WSIL**

Use this table if you want to register services for a referenced WSIL instead of the current WSIL. When you click the Process icon below, the current WSIL on this page will be replaced by the referenced WSIL. You can then register services from the referenced WSIL.

Reference Location	Reference Description	Process
http://link1.com/ecommerce.wsil	Acme Industries Public e-Co...	⚙
http://link2.com/services/ecommerce.wsil	Acme Industries Public e-Co...	⚙

The Register New Service page displays the Source information, in read-only format, and a list of available services, if any, in the WSIL that you can register, shown in the Services Available in WSIL table. If there are any WSIL references in the WSIL, they are listed in the References Available in WSIL table.

In the Services Available in WSIL table, you can edit the details of a service by clicking the **Edit** icon, which allows you to change the name and description of the selected service.

6. To register a service available from the current WSIL, select the service in the Services Available in WSIL table and click **Register**.

   A confirmation message displays indicating that the service was registered successfully.

7. If the current WSIL also references other WSIL URLs or references, expand **References Available in WSIL** to display them. You can register the referenced Web services as well.

   To register a service from a referenced WSIL instead of the current WSIL, click the **Process** icon for the reference in the References Available in WSIL table.

   If the WSIL parses successfully, a new source is registered and the current WSIL source is replaced by the referenced WSIL. The services available in the referenced WSIL source are listed in the Services Available in WSIL table. You can then register services from the referenced WSIL.

   > **Note:** For each new source created, `_n` is appended to the parent source name. For example, if the parent source name is `wsil_file_1`, then referenced new sources are named `wsil_file_1_`*`1`*, `wsil_file_1_`*`2`*) with source type WSIL URL. The new sources are listed in the Sources table in the Registered Sources and Services page.

   If the WSIL does not parse successfully, an error message displays. Usually, in such cases, the system successfully registers the new source for the selected WSIL reference. However, because the system could not parse the WSIL document, the error message displays. Close the error dialog and click **OK** to return to the Registered Sources and Services page.

WSIL parsing can fail if the reference is bad or it needs authorization credentials. You can enable authorization for the WSIL source as described in "Registering a Source" on page 14-4.

> **Note:**
>
> When the system fails to retrieve Web services from a registered source, because of connection or other failures, the Register New Service page is displayed with read only information for the source, but does not show any Web services. In such cases, click **OK** in the error dialog, if an error dialog is displayed, then click **OK** in the Register New Service page to return to the Registered Sources and Services page. To troubleshoot, you can then view the registered sources through other means. For example, if the source is a:
>
> - WSIL URL source, copy the URL to a browser address bar to view its contents.
> - WSIL file source, examine the WSIL file using an XML editor.
> - UDDI source, try to access the UDDI registry directly to investigate.
>
> You can also review any related Enterprise Manager error logs.

14.1.7 Deleting a Web Service or Web Service Source

Follow the steps in this section to delete a Web service or a Web service source.

1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to delete a Web service.
2. Select the domain.
3. Using Fusion Middleware Control, select **WebLogic Domain** then **Web Services** and then **Registered Services**. The Registered Sources and Services page displays, as shown in Figure 14–2.
4. Do one of the following:
   - To delete a source, select the source from the Sources table and click **Delete**.

     A confirmation message displays. Click **OK** to delete the source.
   - To delete a service from a source, select the source in the Sources table.

     The registered Web services are displayed in the Services table.

     Select the service to be deleted from the Services table and click **Delete**.

     A confirmation message displays. Click **OK** to delete the service.

14.2 Publishing Web Services to UDDI

You can publish Web services to UDDI from a registered UDDI source and from the Web services summary page for ADF, WebCenter, and Java EE applications. Registered UDDI sources are listed in the Registered Sources and Services page, which includes all sources and services registered in a domain. The Web services summary page lists the Web services in an application.

Notes: You must use a proxy to publish a service to UDDI, since this requires access to URLs outside of your firewall. For more information about the required proxy settings, see "Configuring the Proxy Server for UDDI" on page 14-11.

If your services are already in Oracle Enterprise Repository (OER) then you should use the OER Exchange Utility to publish those services to Oracle Service Registry.

The following procedures describe how to publish Web services to UDDI.

- "Publishing a Web Service to UDDI from a Registered Source" on page 14-9
- "Publishing a Web Service to UDDI from an Application" on page 14-10
- "Configuring the Proxy Server for UDDI" on page 14-11

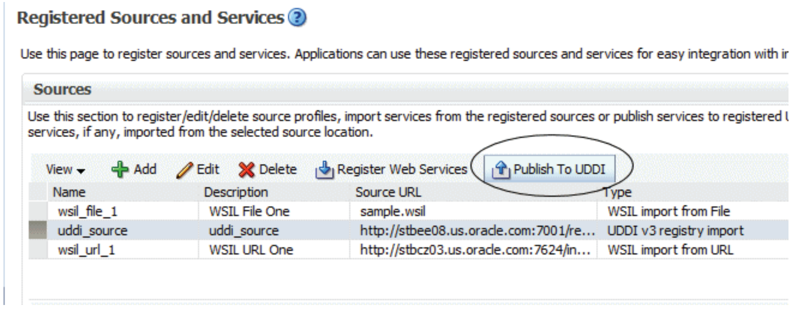
14.2.1 Publishing a Web Service to UDDI from a Registered Source

To publish a Web service to UDDI from a registered source:

1. Navigate to the Registered Sources and Services page as described in "Viewing Registered Sources and Web Services" on page 14-3.

2. Select the source row in the Sources table and then **Publish to UDDI.**

Figure 14–6 Registered Sources and Services Page with Publish to UDDI Selected



Registered Sources and Services

Use this page to register sources and services. Applications can use these registered sources and services for easy integration with ir

Sources

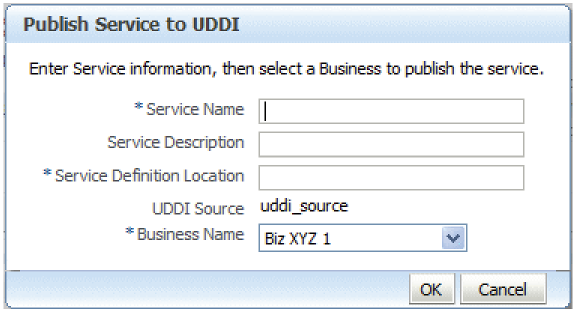
Use this section to register/edit/delete source profiles, import services from the registered sources or publish services to registered U
services, if any, imported from the selected source location.

Name	Description	Source URL	Type
wsil_file_1	WSIL File One	sample.wsil	WSIL import from File
uddi_source	uddi_source	http://stbee08.us.oracle.com:7001/re...	UDDI v3 registry import
wsil_url_1	WSIL URL One	http://stbcz03.us.oracle.com:7624/in...	WSIL import from URL

3. In the Publish Service to UDDI window, enter the information about the service to be published:

   - **Service Name** is the name of the Web service to be published to the UDDI registry. This field is required.
   - **Service Description** is a description of the selected Web service.
   - **Service Definition Location** is the URL location of the service definition. This field is required.
   - **UDDI Source** is the name of the UDDI source from which the service is to be registered. This field is read only.
   - **Business Name** is the name of the data structure in the UDDI registry. It is assumed that the business has already been registered in the UDDI. Choose the Business name from the list. This field is required.

Figure 14–7 Publish Service to UDDI Window from a UDDI Source



Publish Service to UDDI

Enter Service information, then select a Business to publish the service.

* Service Name	
Service Description	
* Service Definition Location	
UDDI Source	uddi_source
* Business Name	Biz XYZ 1

4. Click **OK** in the Publish Service to UDDI window.

   The system verifies that the service specified has a valid WSDL and that the UDDI registry has accepted the new entry or updated an existing one. If it is successful, a confirmation message displays and the service is published to the registry.

   Once the service is published in the UDDI, it becomes available to be registered to a source, as described in "Registering Web Services from a UDDI Source" on page 14-5.

   Any errors during the operation will result in an error message.

   Note that you can only register the service to a source if it uses a unique WSDL.

14.2.2 Publishing a Web Service to UDDI from an Application

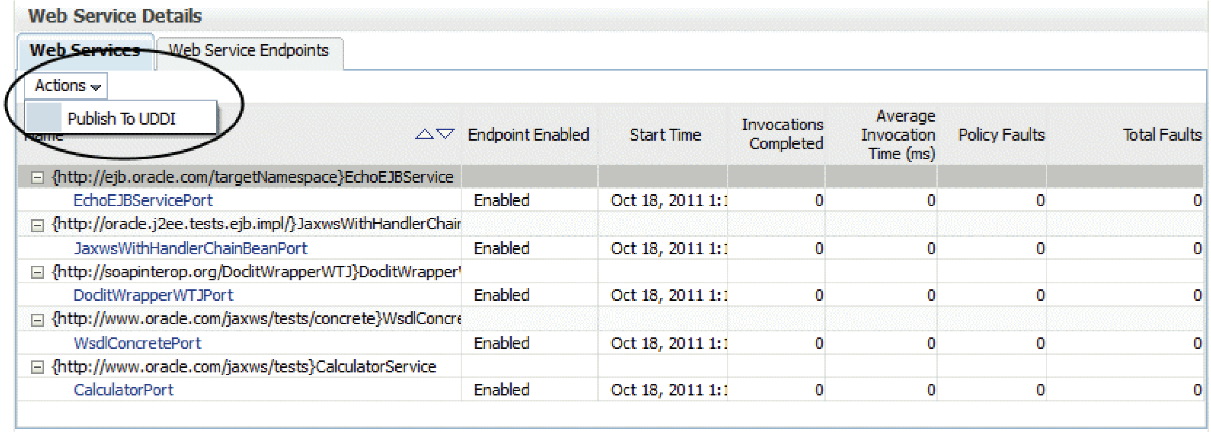
To publish a Web service to UDDI from an application:

1. Navigate to the Web Services summary page as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.

2. From the Web Service Details section of the page, select the service to be published.

3. Select **Actions**, then **Publish to UDDI.** See Figure 14–8.

Figure 14–8 Web Services Summary Page with Publish to UDDI Selected



Name	Endpoint Enabled	Start Time	Invocations Completed	Average Invocation Time (ms)	Policy Faults	Total Faults
⊟ {http://ejb.oracle.com/targetNamespace}EchoEJBService						
EchoEJBServicePort	Enabled	Oct 18, 2011 1::	0	0	0	0
⊟ {http://oracle.j2ee.tests.ejb.impl/}JaxwsWithHandlerChair						
JaxwsWithHandlerChainBeanPort	Enabled	Oct 18, 2011 1::	0	0	0	0
⊟ {http://soapinterop.org/DoclitWrapperWTJ}DoclitWrapper'						
DoclitWrapperWTJPort	Enabled	Oct 18, 2011 1::	0	0	0	0
⊟ {http://www.oracle.com/jaxws/tests/concrete}WsdlConcre						
WsdlConcretePort	Enabled	Oct 18, 2011 1::	0	0	0	0
⊟ {http://www.oracle.com/jaxws/tests}CalculatorService						
CalculatorPort	Enabled	Oct 18, 2011 1::	0	0	0	0

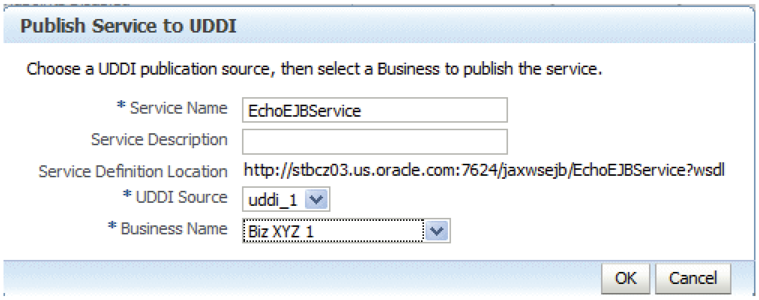
4. In the Publish Service to UDDI dialog box (Figure 14–9), enter the information about the service to be published:

- **Service Name** is the name of the Web service to be published to the UDDI registry. This field is required.
- **Service Description** is a description of the selected Web service.
- **Service Definition Location** is prepopulated with the URL location of the service definition (This field is read-only.)
- **UDDI Source** is a logical name for the UDDI registry source. Choose the UDDI source from the list. This field is required.

  Note: The list contains the UDDI sources registered in the domain that have been enabled for publishing. For more information about registered sources, see "Registering Web Services and Sources" on page 14-1.

- **Business Name** is the name of the data structure in the UDDI registry. It is assumed that the business has already been registered in the UDDI. Choose the business name from the list. This field is required.

Figure 14–9 Publish Service to UDDI Dialog Box



Publish Service to UDDI

Choose a UDDI publication source, then select a Business to publish the service.

* Service Name: EchoEJBService

Service Description:

Service Definition Location: http://stbcz03.us.oracle.com:7624/jaxwsejb/EchoEJBService?wsdl

* UDDI Source: uddi_1

* Business Name: Biz XYZ 1

5. Click **OK** to connect to the external UDDI registry and register the Web service.

   Upon successfully registering the service, a confirmation message displays. Any errors during the operation will result in an error message.

14.2.3 Configuring the Proxy Server for UDDI

To access URLs outside of your firewall, you must use a proxy to publish a service to UDDI.

Before starting Oracle WebLogic, you must set the Java system properties defined in Table 14–1. You can set them as environment variables, or in Oracle WebLogic startup files.

Table 14–1 Java System Properties Used to Specify the Proxy Server for UDDI

Property	Description
proxySet=true	Flag that specifies that the WebLogic proxy properties should be used.
http.proxyHost=*proxyHost*	Name of the host computer on which the proxy server is running.
http.proxyPort=*proxyPort*	Port to which the proxy server is listening.
http.nonProxyHosts=*hostname* | *hostname* | ...	List of hosts that should be reached directly, bypassing the proxy. Separate each host name using a | character.

For example:

```
set PROXY_SETTINGS="-DproxySet=true -Dhttp.proxyHost=www-proxy.example.com
-Dhttp.proxyPort=80 -Dhttp.nonProxyHosts=localhost|${HOST}|*.example.com"
```

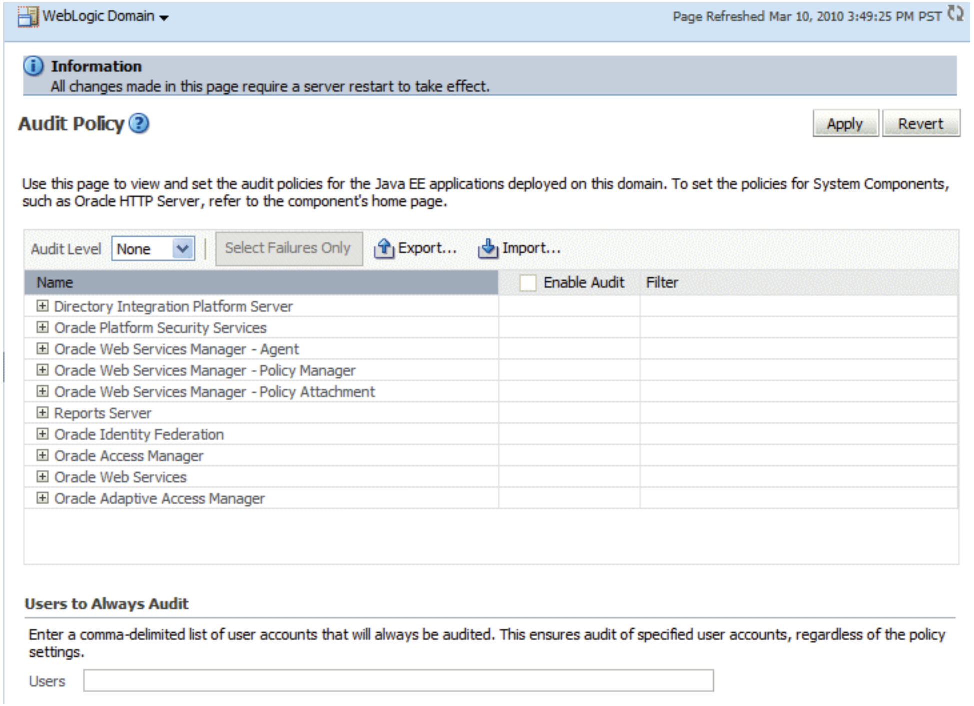
14.3 Auditing Web Services

Auditing describes the process of collecting and storing information about security events and the outcome of those events. An audit provides an electronic trail of selected system activity.

An audit *policy* defines the type and scope of events to be captured at run time. Although a very large array of system and user events can occur during an operation, the events that are actually audited depend on the audit policies in effect at run time. You can define component- or application-specific policies, or audit individual users.

You configure auditing for system components, including Web services, and applications at the domain level using the Audit Policy page. You can audit SOA, ADF, and WebCenter services.

Figure 14–10 Audit Policy Page



WebLogic Domain ▾

Page Refreshed Mar 10, 2010 3:49:25 PM PST

Information
All changes made in this page require a server restart to take effect.

Audit Policy

Apply | Revert

Use this page to view and set the audit policies for the Java EE applications deployed on this domain. To set the policies for System Components, such as Oracle HTTP Server, refer to the component's home page.

Audit Level: None | Select Failures Only | Export... | Import...

Name	□ Enable Audit	Filter
⊞ Directory Integration Platform Server		
⊞ Oracle Platform Security Services		
⊞ Oracle Web Services Manager - Agent		
⊞ Oracle Web Services Manager - Policy Manager		
⊞ Oracle Web Services Manager - Policy Attachment		
⊞ Reports Server		
⊞ Oracle Identity Federation		
⊞ Oracle Access Manager		
⊞ Oracle Web Services		
⊞ Oracle Adaptive Access Manager		

Users to Always Audit

Enter a comma-delimited list of user accounts that will always be audited. This ensures audit of specified user accounts, regardless of the policy settings.

Users

The audit policies table, at the center of the page, displays the audits that are currently in effect. The table includes the following information:

- Name—Name of the system components and applications that you can audit.
- Enable Audit—Identifies the components and applications for which auditing is currently in effect.
- Filter—Specifies any filters that are currently in effect.

The following table summarizes the events that you can audit for Web services and the relevant component.

Table 14–2 Auditing Events for Web Services

Enable auditing for the following Web service events . . .	Using this system component . . .
■ User authentication. ■ User authorization. ■ Policy enforcement, including message confidentiality, message integrity, and security policy.	Oracle Web Services Manager—Agent
■ Web service requests sent and responses received. ■ SOAP faults incurred.	Oracle Web Services
■ Oracle WSM assertion template creation, deletion, or modification. ■ Oracle WSM policy creation, deletion, or modification. ■ Oracle WSM policy set authoring creation, deletion, or modification.	Oracle Web Services Manager—Policy Manager **Note:** The Policy Manager audits both direct policy attachments and global policy attachments for policy sets.
■ Oracle WSM policy attachment.	Oracle Web Services Manager—Policy Attachment **Note:** The Policy Attachment audits only direct policy attachments.

You can also audit the events for a specific user, for example, you can audit all events by an administrator.

For more information about configuring audit policies, see "Configuring and Managing Auditing" in *Oracle Fusion Middleware Application Security Guide*.

The following sections describe how to define audit policies and view audit data:

- Configuring Audit Policies
- Managing Audit Data Collection and Storage
- Viewing Audit Reports

14.3.1 Configuring Audit Policies

Follow the steps in this section to configure audit policies.

1. In the Navigator pane, expand **WebLogic Domain**.
2. Click the domain for which you want to manage assertion templates.
3. From the WebLogic Domain menu select **Security > Audit Policy**.

   The Audit Policy Settings page is displayed.
4. Select and audit level from the Audit Level menu.

   Valid audit levels include:

   - None—Disables auditing.
   - Low—Audits a small scope of events. The subset of events is predefined individually for each component. For example, for a given component, Low may collect authentication and authorization events only.

- Medium—Audits a medium scope of events (which is a superset of the events collected at the Low level). For example, for a given component, Medium may collect authentication, authorization, and policy authoring events.
- Custom—Enables you to provide a custom auditing policy.

You can view the components and applications that are selected for audit at each level in the audit policies list. For all audit levels other than Custom, the information in the audit policies list is greyed out, as you cannot customize other audit level settings.

5. If you selected the Custom audit level, perform one of the following steps:

   - Select the information that you want to audit by clicking the associated check box in the Enable Audit column.

     You can audit at the following levels of granularity: All events for a component, all events within a component event group, an individual event, or a specific outcome of an individual event (such as success or failure).

     At the event outcome level, you can specify an edit filter. Filters are rules-based expressions that you can define to control the events that are returned. For example, you might specify an Initiator as a filter for policy management operations to track when policies were created, modified, or deleted by a specific user. To define a filter for an outcome level, click the **Edit Filter** icon in the appropriate column, specify the filter attributes, and click **OK**. The filter definition appears in the Filter column.

     Deselect the check box for a component at a higher level to customize auditing for its subcomponents. You can select all components and applications by checking the check box adjacent to the column name.

   - To audit only failures for all system components and applications, **Select Failures Only**.

     If selected, all checkboxes in the Enable Audit column are cleared.

6. If required, enter a comma-separated list of users in the Always Audit Users text box.

   Specified users will always be audited, regardless of whether auditing is enabled or disabled, and at what level auditing is set.

7. Click **Apply.**

   To revert all changes made during the current session, click **Revert**.

14.3.2 Managing Audit Data Collection and Storage

To manage the data collection and storage of audit information, you must perform the following tasks:

- Set up and manage an audit data repository.

  You can store records using one of two repository modes: file and database. It is recommended that you use the database repository mode. The Oracle Business Intelligence Publisher-based audit reports only work in the database repository mode.

- Set up audit event collection.

For more information, see "Managing the Audit Store" in *Oracle Fusion Middleware Application Security Guide*.

14.3.3 Viewing Audit Reports

For database repositories, data is exposed through pre-defined reports in Oracle Business Intelligence Publisher.

A number of predefined reports are available, such as: authentication and authorization history, Oracle WSM policy enforcement and management, and so on. For details about generating and viewing audit reports using Oracle Business Intelligence Publisher, see "Using Audit Analysis and Reporting" in *Oracle Fusion Middleware Application Security Guide*.

For file-based repositories, you can view the bus-stop files using a text editor and create your own custom queries.

14.4 Managing the WSDL

In some cases, you might not want the Web service WSDL to be accessible to the public. You can enable or disable public access to the WSDL from the Web Service Endpoint page.

> **Note:** In some cases, a Web service client needs to access a WSDL during invocation. If public access to the WSDL is disabled, the client must have a local copy of the WSDL.

To manage the WSDL:

1. Navigate to the Web Service endpoint configuration page, as described in "Configuring the Web Service Endpoint" on page 6-13.
2. On the Configuration tab, set the WSDL Enabled field to **True** or **False** to enable of disable public access to your WSDL, respectively.
3. Click **Apply.**

14.5 Adding Security to a Running Client

Security policies can be attached to a running client using Oracle Enterprise Manager Fusion Middleware Control. You do not have to redeploy the client application to attach or detach policies from the client. See Chapter 8, "Attaching Policies to Web Services" for more information on how to attach policies using Fusion Middleware Control.

14.6 Configuring Platform Policy Properties

You can manage properties for the following components from the Platform Policy Configuration page:

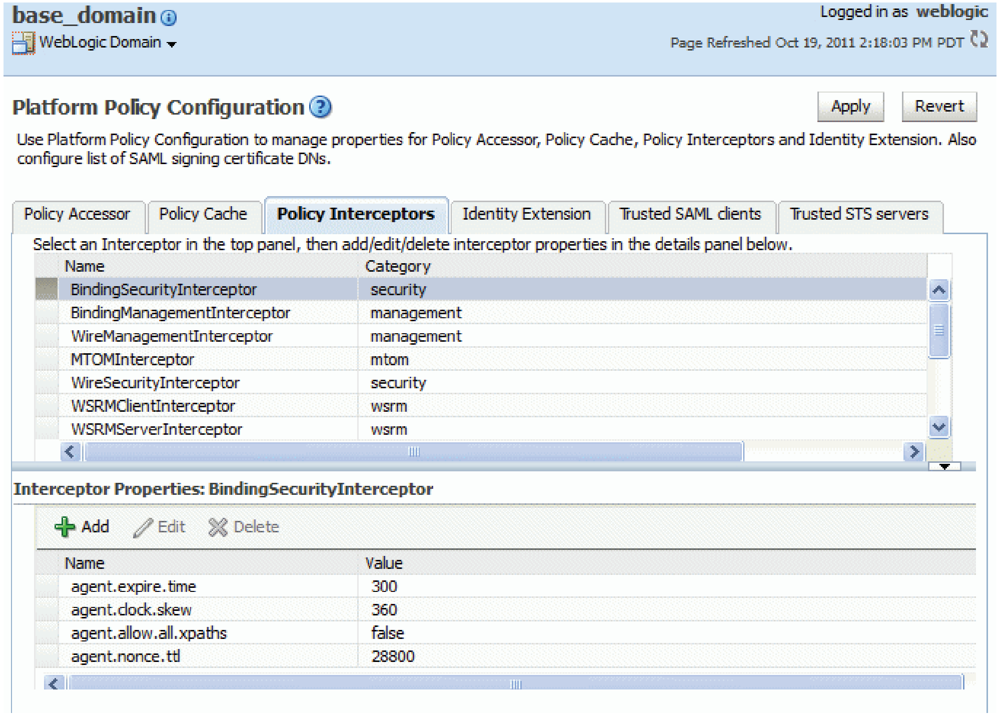
- Policy Accessor
- Policy Cache
- Policy Interceptors
- Identity Extension
- Trusted SAML clients
- Trusted STS servers

To manage policy accessor, policy cache, policy interceptor, and identity extension properties:

1. In the navigator pane, expand **WebLogic Domain** to view the domains.
2. Select the domain for which you want to manage properties.
3. Select **WebLogic Domain> Web Services > Platform Policy Configuration**.

   The Platform Policy Configuration page appears, as shown in Figure 14–11.

Figure 14–11 Platform Policy Configuration Page



Use Platform Policy Configuration to manage properties for Policy Accessor, Policy Cache, Policy Interceptors and Identity Extension. Also configure list of SAML signing certificate DNs.

Select an Interceptor in the top panel, then add/edit/delete interceptor properties in the details panel below.

Name	Category
BindingSecurityInterceptor	security
BindingManagementInterceptor	management
WireManagementInterceptor	management
MTOMInterceptor	mtom
WireSecurityInterceptor	security
WSRMClientInterceptor	wsrm
WSRMServerInterceptor	wsrm

Name	Value
agent.expire.time	300
agent.clock.skew	360
agent.allow.all.xpaths	false
agent.nonce.ttl	28800

4. Select the tab corresponding to the component for which you want to define properties:

   - "Configuring the Policy Manager Connection and Tuning the Policy Cache" on page 14-16
   - "Configuring Web Service Policy Retrieval" on page 14-18
   - "Tuning Web Service Security Policy Enforcement" on page 14-20
   - "Defining Identity Extension Properties" on page 14-22
   - "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23

14.7 Configuring the Policy Manager Connection and Tuning the Policy Cache

By default, the Oracle Web Services Manager (WSM) supports an auto-discovery feature that it uses to locate and connect to an Oracle WSM Policy Manager within the same WebLogic domain. In certain scenarios auto-discovery may not work as expected.

> **Note:** When the Oracle WSM Policy Manager is deployed on a server that is configured to use SSL, the auto-discovery mechanism will only attempt to connect to Policy Managers on other SSL-configured servers. To ensure that the secure connection is maintained, Policy Managers deployed on servers that are not configured for SSL are ignored.

You may want to disable the auto-discovery feature, for example, in the following scenarios:

- Your domain is split into two or more networks, especially if a firewall exists between them.
- You are running on a non-WebLogic application server that does not support the auto-discovery feature, such as WebSphere Application Server.
- You prefer to override the default settings.

For Oracle Infrastructure Web service policies, on the Platform Policy Configuration page:

- The **Policy Accessor** tab enables you to specify an alternate Policy Manager URL and corresponding credentials to access the Policy Manager.
- The **Policy Cache** tab allows you to tune the behavior of the policy cache delay for Web service endpoints, which can help to avoid network calls and increase performance when fetching policies from the Policy Manager.

To configure a Policy Manager connection and tune the policy cache:

1. Access the Platform Policy Configuration page, as described in "Configuring Platform Policy Properties" on page 14-15.
2. Select the **Policy Accessor** tab.
3. Click **Add** to define a JNDI provider.

   In the Add Property window, specify the following values:

   a. In the **Name** field, enter the JNDI provider URL property as `java.naming.provider.url`.

   b. In the **Value** field, enter the JNDI provider's URL, for example `t3://`*`host`*`:`*`port`*.

   c. Click **OK**.
4. Click **Add** to define a corresponding csf-key credential property. In the Add Property window, specify the following values:

   a. In the **Name** field, enter the name of the JNDI provider's csf-key credential property as `jndi.lookup.csf.key`.

   b. In the **Value** field, enter the csf-key credentials.

   Because the Policy Manager is security enabled, the csf-key specifies the `java.naming.security.principal` and `java.naming.security.credentials` when using the JNDI URL to look up a Policy Manager.

   c. Click **OK**.

   For more information on storing, retrieving, and deleting credentials, see "Adding Keys and User Credentials to the Credential Store" on page 10-19

5. Select the **Policy Cache** tab.

6. To modify the policy cache property for Web service endpoints, select it and then click **Edit**. In the Edit Property window, you can edit the **Value** field to change the default amount for each property.

   a. `cache.tolerance` – Amount of time (in milliseconds) between refreshes of the policy cache. This ensures that the policy set retrieved from the Web service endpoint policy cache is the most current version (that is, it has not exceeded the `cache.tolerance` value). If it is determined that the policy set is stale, the updated policy set is retrieved from the Oracle WSM Policy Manager and refreshed in the Web service endpoint policy cache. The default is 60000 milliseconds (1 minute).

      Note: The refresh delay amount for Web service endpoints is aggregated with the value of the `cache.refresh.repeat` property on the Policy Accessor tab for the Oracle WSM Policy Manager. Therefore, you should verify whether this additional value produces the desired refresh delay when combined with the `cache.refresh.repeat` amount. For more information, see "Tuning WSM Repository Connections" on page 14-19.

   b. To add another property, click **Add**, and in the Add Property window, specify the necessary values.

   c. Click **OK**.

7. To modify an existing property, select it and then click **Edit**.

8. To delete an existing property, select it and then click **Delete**.

9. Click **Apply** to apply the property updates.

14.8 Configuring Web Service Policy Retrieval

The Policy Accessor tab also enables you to configure the retrieval of Oracle WebLogic Web service policies from a repository. This includes specifying the repository location (JARs, directory, or host and port) and the account information.

1. Access the Platform Policy Configuration page, as described in "Configuring Platform Policy Properties" on page 14-15.

2. Select the **Policy Accessor** tab.

3. Click **Add** to add a policy retrieval property.

4. Use the following table to specify the property names and values in the Add New Configure Property window:

Table 14–3 Properties in Add Property Window

Element	Description
java.naming.provider.url	JNDI URL that specifies the location of a running Oracle WSM Policy Manager, for example `t3://`*`host:port`*. By default, this property is not specified. If this property is not specified, Oracle WSM auto-discovery attempts to look up the Policy Manager in the same domain.

Table 14–3 (Cont.) Properties in Add Property Window

Element	Description
jndi.lookup.csf.key	If the location of the Oracle WSM Policy Manager is provided in the `java.naming.provider.url` property, the `jndi.lookup.csf.key` provides credential configuration. Because the Oracle WSM Policy Manager is security enabled, the `jndi.lookup.csf.key` specifies the `java.naming.security.principal` and `java.naming.security.credentials` when using the JNDI URL to look up a Oracle WSM Policy Manager. By default, this property is not specified. You should configure this property when: ■ You want to specify an explicit account to connect with the Oracle WSM Policy Manager rather than the system account, `OracleSystemUser`, that is used by Oracle WSM by default. ■ The Authentication Provider and LDAP directory that is configured does not support system accounts used by Oracle WebLogic, but which Oracle WSM relies on by default. Therefore, a different account in the LDAP directory must be used. ■ There is no concept of default system accounts in a particular application server, and so the system cannot rely on system accounts. For more information on modifying the default user, see "Modifying the Default User" on page 14-37.

5. To modify an existing property, select it and then click **Edit**.
6. To delete an existing property, select it and then click **Delete**.
7. Click **Apply** to apply the property updates.

14.9 Tuning WSM Repository Connections

The properties on the Policy Accessor tab also enable you to configure the connection between the Agent and the Oracle WSM Policy Manager, including retry logic (for high availability), and cache refresh rates.

The configuration management system will periodically reconnect to the Policy Manager (for example, to handle situations when the connection information changes). If the Policy Manager is down when the runtime attempts to reconnect, then it will use the value of the `connect.retry.delay` property to determine when it tries again.

If the initial connection is made, but the Oracle WSM Repository is not operating properly, then services will start in a "non-operational" state. You can adjust the values of the `failure.retry.count` and the `failure.retry.delay` properties to determine how may times the Agent will attempt to communicate with the Policy Manager, which in turn accesses the repository, and the time interval between retries. When the repository becomes available, then the services will become operational.

The `cache.refresh.initial` and `cache.refresh.repeat` properties can be adjusted to affect how often the Agent attempts to contact the Policy Manager to refresh documents that it has already cached. The `missing.retry.delay` property indicates how often the Agent will attempt to connect to the Policy Manager to retrieve a document that it could not retrieve. If a document or group of related documents (such as policy sets) cannot be retrieved because communication with the Policy Manager fails (for example, because it went down), then the `missing.retry.delay` property will still affect how often it attempts to communicate with the Policy Manager until it is fixed.

To view or change the settings for these properties:

1. Access the Platform Policy Configuration page, as described in "Configuring Platform Policy Properties" on page 14-15.
2. Select the **Policy Accessor** tab.
3. Select the property in the table and click **Edit**.

   Table 14–4 lists the available properties and their default settings.

Table 14–4 Properties for Tuning WSM Repository Connections

Property	Description	Default
`connect.retry.delay`	Number of milliseconds to wait before the Agent attempts to establish a connection to the Policy Manager after a failure.	600000 milliseconds (10 minutes)
`cache.refresh.initial`	Number of milliseconds to wait before initial cache refresh.	600000 milliseconds (10 minutes)
`cache.refresh.repeat`	Number of milliseconds to wait between cache refreshes.	600000 milliseconds (10 minutes)
`missing.retry.delay`	Number of milliseconds to wait before trying to retrieve a missing document.	15000 milliseconds
`usage.record.delay`	Number of milliseconds to wait before sending usage data.	30000 milliseconds
`failure.retry.count`	Number of times to retry after communication failure.	2 retry attempts
`failure.retry.delay`	Number of milliseconds to wait between retry attempts. The default is 5000 milliseconds.	

4. Enter the changes required and click **OK**.
5. Edit any remaining properties as desired and then click **Apply** to apply the property updates.

14.10 Tuning Web Service Security Policy Enforcement

The `BindingSecurityInterceptor` property on the **Policy Interceptors** tab allows you to tune security policy enforcement by adjusting the default message timestamp skews between system clocks, the time-to-live for nonce messages in the policy cache, and the message expiration time.

Perform the following steps to tune the security policy enforcement:

1. Access the Platform Policy Configuration page, as described in "Configuring Platform Policy Properties" on page 14-15.
2. Select the **Policy Interceptors** tab.
3. Select the `BindingSecurityInterceptor` security property on the list.
4. Table 14–5 list the properties that you can set to tune Web service security policy enforcement.

To modify a BindingSecurityInterceptor security property, select it and then click **Edit**. In the Edit Property window, you can edit the **Value** field to change the default amount for each property and click **OK**.

Table 14–5 Properties for Tuning Web Service Security Policy Enforcement

Property	Description	Default
agent.clock.skew	Tolerance of time differences, in seconds, between client and server machines. For example, when timestamps are sent across in a message to a service that follows a different time zone, this property allows for the specified time tolerance. Increase agent.clock.skew when: ■ The server's clock is ahead of the client's clock. If the server's clock is ahead of the client's clock then increase the agent.clock.skew. For example, if the server's clock is ahead of the client's clock by 10 minutes, then increase the server's agent.clock.skew to 10 minutes. ■ The client's clock is ahead of the server's clock. If the client's clock is ahead of the server's clock then increase the agent.clock.skew. For example, if the client's clock is ahead of the server's clock by 10 minutes, then increase the server's agent.clock.skew to 10 minutes.	360 seconds (6 minutes)
agent.client.clock.skew	Tolerance of time, in seconds, that is used to calculate the NotBefore and NotOnOrAfter conditions for SAML or JWT token generation. Together, these conditions define the lower and upper boundaries to limit the validity of the token. **Note**: This property does not appear in the BindingSecurityInterceptor Interceptor Properties table by default. To modify the property value, add the property to the table by clicking **Add**, setting the **Name** field to **agent.client.clock.skew** and the **Value** field to the desired value, and clicking **OK**.	0 seconds

Table 14–5 (Cont.) Properties for Tuning Web Service Security Policy Enforcement

Property	Description	Default
`agent.nonce.ttl`	Total time-to-live, in seconds, for nonce in the cache when nonce is sent across in a message. This property caches the nonce and once this duration is over, the nonce is removed from the cache. **Note:** For username token settings, you must enable both the Creation Time Required and Nonce Required properties to prevent replay attacks. If only Nonce Required is enabled, then nonce will be cached forever to prevent replay attacks. Additionally, you must set the value of `agent.nonce.ttl` to be equal to or greater than the value set for `agent.expire.time`.	`28800` seconds (8 hours)
`agent.expire.time`	Duration of time, in seconds, before a message expires after its creation. This property is used in cases where a timestamp is sent across in the SOAP header to verify if the timestamp has expired or not. If the message expires when received by the service even when there is no time difference between the client's and service's clocks, then the message expiry time must be increased. The message expiry time is derived from the values of `agent.expire.time` and the expiry time in the incoming message, and is the lesser of the two. For example, if the server's `agent.expire.time` is set to 5 minutes and expiry time in the incoming message expiry time is 6 minutes, then the `agent.expire.time` at the service side must be increased. On the other hand, if the server's `agent.expire.time` is 5 minutes and the incoming message expiry time is 3 minutes, then the expiry time in the incoming message (that is, at the client side) must be increased. A higher value of the `agent.expire.time` may lead to a security vulnerability.	`300` seconds (5 minutes)
`agent.allow.all.xpaths`	Property that specifies whether Oracle WSM will accept all types of XPath transformations. By default, Oracle WSM only accepts the XPath transformation `ancestor-or-self::*[namespace-uri()='<namespace>'` and `local-name()='<name>']` inside the signature (in the incoming soap message). Set this property to `true` to allow and accept all types of XPath transformations. **Note**: Enabling this property may result in XPath-based Denial of Service attacks or other similar XPath based security vulnerabilities.	`false`

5. To delete an existing property, select it and then click **Delete**.
6. Click **Apply** to apply the property updates.

14.11 Defining Identity Extension Properties

The properties on the Identity Extension tab enable you to specify whether to enforce Web service policies by publishing the X509 certificate in the WSDL. If there is no need to publish the X509 certificate (for example, with SSL), you can override the default setting to avoid publishing the certificate. In addition, if the X509 certificate is published, you can also specify whether to ignore the hostname verification feature.

For more information, see "Using Service Identity Certification Extension" on page 10-57.

14.12 Defining Trusted Issuers and a Trusted DN List for Signing Certificates

In Fusion Middleware Control, the Trusted SAML clients, Trusted JWT clients, and Trusted STS servers tabs enable you to define SAML and JWT trusted issuers and a list of trusted distinguished names (DNs) for SAML and JWT signing certificates.

> **Note:** You can define SAML and JWT trusted issuers, using WLST as described in "Defining Trusted Issuers and Managing DN Lists Using WLST" on page 14-25 or using the REST API, as described in "Trust Configuration Management" in *REST API for Managing Credentials and Keystores with Oracle Web Services Manager*.

This is the preferred method of adding trusted issuers. The list of trusted issuers that you define here becomes the default list that is applicable to all Web services in this domain. In addition, when you add an issuer using this method, it does not require a restart of the domain.

> **Note:** There is a hierarchy that determines how trusted issuers are determined:
>
> 1. The list of trusted issuers configured for policies (or overridden) is checked and used, as specified by the `saml.trusted.issuers` property for the policy. For example, see `saml.trusted.issuers` in Table C–28, " wss10_saml20_token_service_template Configurations".
> 2. If not configured for policies (or overridden), the configuration at the platform policy configuration as described in this section is checked and used.
> 3. If the list of trusted issuers is not configured for policies (or overridden) or configured at the platform policy configuration, only then is the Issuers list defined in the SAML login module used. For more information about the Issuers list defined in the SAML login module, see Step 4 in "Configuring the SAML and Kerberos Login Modules" on page 10-60.

By default, Oracle WSM checks the incoming issuer name against the list of configured issuers, and checks the SAML and JWT signatures against the configured certificates in the Oracle WSM keystore. If you define a trusted DNs list, Oracle WSM also verifies that the SAML and JWT signatures are signed by the particular certificate(s) that is associated with that issuer.

Configuration of the trusted DNs list is optional; it is available for users that require more fine-grained control to associate each issuer with a list of one or more signing certificates. If you do not define a list of DNs for a trusted issuer, then Oracle WSM allows signing by any certificate, as long as that certificate is trusted by the certificates present in the Oracle WSM keystore.

Important Notes:

- Using the Trusted SAML clients, Trusted STS servers, and Trusted JWT clients tabs, or the associated WLST commands, you define the DNs of the *signing certificates*, not the certificates themselves.
- The certificate must be imported into the Oracle WSM keystore or included in the message. If the certificate is provided in the message, its issuer must be imported into the Oracle WSM keystore.
- For two-way SSL:

  - The certificate needs to be imported into the Java EE container's trust store.
  - The DN of the client SSL certificates are used for validation and must be present in the trusted DNs list.
- In all cases, the signing certificates must be trusted by the certificates present in the OWSM keystore.

You can define trusted issuers and DN lists using Fusion Middleware Control or WLST.

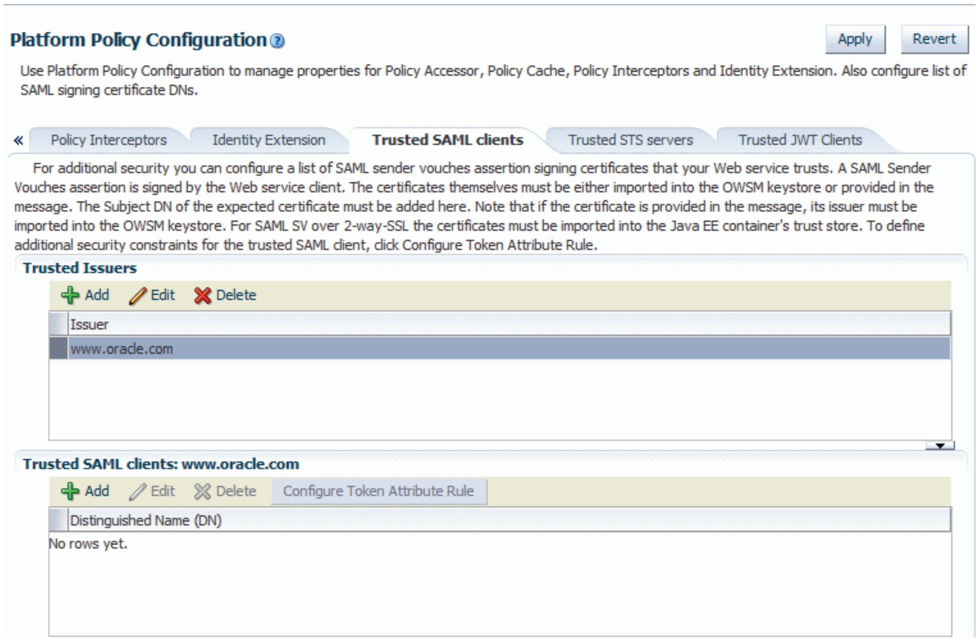
- Configuring an Issuer and its DN List Using Fusion Middleware Control
- Configuring an Issuer and its DN List Using WLST

14.12.1 Configuring an Issuer and its DN List Using Fusion Middleware Control

To define a trusted DN list for a SAML signing certificate:

1. Access the Platform Policy Configuration page, as described in "Configuring Platform Policy Properties" on page 14-15.
2. Select the **Trusted SAML clients**, **Trusted STS servers**, or **Trusted JWT clients** tab, depending on whether you want to define a trusted DNs list for trusted SAML clients (for SAML sender vouches), trusted STS servers (for SAML HOK and SAML bearer), or trusted JWT clients (for JWT issuers.) See Figure 14–12.

Figure 14–12 Adding Trusted Issuers on the Platform Policy Configuration Page



Platform Policy Configuration

Use Platform Policy Configuration to manage properties for Policy Accessor, Policy Cache, Policy Interceptors and Identity Extension. Also configure list of SAML signing certificate DNs.

For additional security you can configure a list of SAML sender vouches assertion signing certificates that your Web service trusts. A SAML Sender Vouches assertion is signed by the Web service client. The certificates themselves must be either imported into the OWSM keystore or provided in the message. The Subject DN of the expected certificate must be added here. Note that if the certificate is provided in the message, its issuer must be imported into the OWSM keystore. For SAML SV over 2-way-SSL the certificates must be imported into the Java EE container's trust store. To define additional security constraints for the trusted SAML client, click Configure Token Attribute Rule.

Trusted Issuers

Issuer
www.oracle.com

Trusted SAML clients: www.oracle.com

Distinguished Name (DN)
No rows yet.

3. Add one or more trusted issuers within the Trusted Issuers section of the page.

   The default value is `www.oracle.com`.

   Click **Add** to add a new trusted issuer. For example: `www.example.com`.
4. Select the trusted issuer for which you want to define a DN list in the Trusted Issuers section of the page.
5. Click **Add** to add one or more trusted DNs in the Trusted SAML clients, Trusted STS servers, or Trusted JWT clients area of the page. Use a string that conforms to

RFC 2253. For example: `CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US` for the trusted issuer `www.oracle.com`.

For more information about RFC 2253, see `http://www.ietf.org/rfc/rfc2253.txt`.

14.12.2 Defining Trusted Issuers and Managing DN Lists Using WLST

The following sections describe how to use WLST commands to define trusted issuers and a list of trusted distinguished names (DNs) for SAML signing certificates and JWT tokens, and how to display and manage the lists associated with a trusted issuer.

- Configuring an Issuer and its DN List Using WLST
- Displaying Issuers and DN Lists using WLST
- Deleting an Issuer and its DN List using WLST
- Importing Trust Metadata Using WLST
- Exporting Trust Metadata Using WLST
- Revoking Trust from Trusted Issuers Using WLST

For more information about these WLST commands, see "Web Services Custom WLST Commands" in the *WebLogic Scripting Tool Command Reference*.

14.12.2.1 Configuring an Issuer and its DN List Using WLST

To configure trusted issuers and DN lists using WLST:

1. Connect to the running instance of the server in the domain for which you want to configure the trusted issuers as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Add the trusted issuers and define trusted keys or a trusted DN list using the `setWSMTokenIssuerTrust` command.

```
setWSMTokenIssuerTrust(type, issuer, [trustedKeyIds=None])
```

In this command:

- `type` indicates the types of the tokens issued by the issuer and how the issuer signing certificates are identified with `trustedKeyIds`. Supported type values are shown in the following table.

Use this type value...	For this token type...	With this key type...	And this key identifier type
`dns.sv`	SAML SV	X509 certificate	DN
`dns.hok`	SAML HOK or Bearer	X509 certificate	DN
`dns.jwt`	JWT	X509 certificate	DN

- `issuer` is the name of the trusted issuer, such as `www.oracle.com`.
- `trustedKeyIds` is an optional argument used to specify the trusted key identifiers or DN list for the issuer.

This command behaves as follows:

- If the trusted issuer already exists for the type specified, and you provide a list of DNs for the `trustedKeyIds` argument, the previous list is replaced with the new list. If you enter an empty set ([]) for the trustedKeyIds argument, then the list of DN values are deleted for the issuer.

- If the trusted issuer does not exist for the type specified and you specify a value for the trustedKeyIds argument, the issuer is created with the associated DN list. If you do not set the trustedKeyIds argument, a new issuer is created with an empty DN list.

In Example 14–1, www.example.com is set as a trusted issuer for a SAML SV token type. A DN list is not specified.

Example 14–1 Setting a Trusted Issuer for SAML Sender Vouches Token Using WLST

```
wls:/base_domain/serverConfig>
setWSMTokenIssuerTrust("dns.sv","www.example.com",[])

the trusted DN lists are successfully set
```

In Example 14–2, CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US' and CN=orcladmin, OU=Doc, O=Oracle, C=US' are set as DNs in the dns.sv DN list for the www.oracle.com trusted SAML issuer.

Example 14–2 Setting DNs for SAML Trusted Issuer Using WLST

```
wls:/base_domain/serverConfig> setWSMTokenIssuerTrust('dns.sv','www.oracle.com',
['CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle',
'CN=orcladmin, OU=Doc, O=Oracle, C=US'])

the trusted DN lists are successfully set
```

In Example 14–3, www.example.com is set as a trusted issuer for a JWT token type with a trusted DN of CN=weblogic1, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US'.

Example 14–3 Setting a Trusted Issuer and DN list for JWT

```
wls:/base_domain/serverConfig> setWSMTokenIssuerTrust('dns.jwt','www.example.com',
['CN=weblogic1, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US'])

JWT trusted issuers successfully set
```

3. Optionally, you can add additional security constraints by defining token attribute rules for a trusted DN. For more information, see "Configuring Token Attribute Rules for Trusted Issuers Using WLST" on page 14-31.

14.12.2.2 Displaying Issuers and DN Lists using WLST

To display the trusted issuers and DN lists using WLST:

1. Connect to the running instance of the server in the domain for which you want to display the trusted issuers as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Display the trusted issuer and DN list using the displayWSMTokenIssuerTrust command.

   ```
   displayWSMTokenIssuerTrust(type, issuer=None)
   ```

   When you specify a value for the type and issuer arguments, the DN lists for the issuer are displayed. If you do not specify an issuer name, all of the trusted issuers for the given type are listed. Supported values for the `type` argument are `dns.sv`, `dns.hok`, and `dns.jwt`.

For example, to view all of the trusted issuers for the type `dns.sv`:

```
wls:/base_domain/serverConfig> displayWSMTokenIssuerTrust('dns.sv')

Starting Operation displayWSMTokenIssuerTrust ...
www.example.com
www.oracle.com
```

To view the DN lists for the `www.oracle.com` trusted issuer for the type `dns.sv`:

```
wls:/base_domain/serverConfig> displayWSMTokenIssuerTrust('dns.sv',
'www.oracle.com')

Starting Operation displayWSMTokenIssuerTrust ...
CN=weblogic, OU=Orakey Test Encryption purposes only, O=Oracle
CN=orcladmin, OU=Doc, O=Oracle, C=US
```

14.12.2.3 Deleting an Issuer and its DN List using WLST

You can delete a trusted issuer and its DN list by using the `deleteWSMTokenIssuerTrust` WLST command:

```
deleteWSMTokenIssuerTrust(issuer, type)
```

Supported values for the `type` argument are `dns.sv`, `dns.hok`, and `dns.jwt`.

In the following example, the issuer `www.example.com` plus the DN lists in the `dns.sv` trusted SAML sender vouches client list for the issuer, if any, are deleted:

```
wls:/base_domain/serverConfig> deleteWSMTokenIssuerTrust('dns.sv',
'www.example.com')
```

14.12.2.4 Importing Trust Metadata Using WLST

You can import trust configurations (issuers, DNs, and token attribute rules) for all trusted issuers from an XML file by using the `importWSMTokenIssuerTrustMetadata` WLST command:

```
importWSMTokenIssuerTrustMetadata(trustFile)
```

In the following example, all trusted issuer configurations are imported from the specified XML file:

```
wls:/base_domain/serverConfig>
importWSMTokenIssuerTrustMetadata(trustFile='/tmp/trustData.xml')

Starting Operation importWSMTokenIssuerTrustMetadata ...
Configuration for trusted issuers successfully imported.
```

14.12.2.5 Exporting Trust Metadata Using WLST

You can export all the trust configurations (issuer, DNs, and token attribute rules) for all trusted issuers to an XML file by using the `exportWSMTokenIssuerTrustMetadata` WLST command:

```
exportWSMTokenIssuerTrustMetadata(trustFile,excludeIssuers=None)
```

The trust configuration for the issuers specified in the exclude list will not be exported. If no argument is passed, the trust configuration for all trusted issuers will be exported.

In the following example, all trusted issuer configurations are exported to the specified XML file except for `www.oracle.com` and `www.example.com`, which have been excluded:

```
wls:/base_domain/serverConfig>
exportWSMTokenIssuerTrustMetadata(trustFile='/tmp/trustData.xml',['www.oracle.com'
,'www.myissuer.com'])

Starting Operation exportWSMTokenIssuerTrustMetadata ...
Configuration for trusted issuers successfully exported.
```

14.12.2.6 Revoking Trust from Trusted Issuers Using WLST

You can remove all trusted issuers and associated configurations (DNs and token attribute rules) by using the `revokeWSMTokenIssuerTrust` WLST command:

```
revokeWSMTokenIssuerTrust(excludeIssuers=None))
```

The issuers specified in the optional exclude list will not be removed. If no argument is passed, then all trusted issuers and the associated configuration are removed.

In the following example, all trusted issuer configurations are removed except for `www.oracle.com` and `www.example.com`, which have been excluded:

```
revokeWSMTokenIssuerTrust(['www.oracle.com','www.example.com'])

Starting Operation revokeWSMTokenIssuerTrust ...
Configuration for trusted issuers successfully removed.
```

14.12.3 Configuring Token Attribute Rules for Trusted Issuers

There are increasing requirements to control which users and user attributes are accepted and processed for a particular trusted user. Token attribute rules allow you to define additional security constraints for the trusted STS (Secure Token Service) server and for the trusted SAML or JWT client.

Token attribute rules can be defined on top of a trusted DN list. For each trusted DN configured for an issuer, a token attribute rule can be configured and applied.

Each rule has two parts: a name ID and an attributes part for attributes associated with a SAML assertion. The name ID and each attribute can contain a filter with multiple value patterns. There is also an optional token attribute mapping capability for federated environments, where the user subject ID (for example, `mail`) in the token is different from the user attribute (for example, `uid`) for authenticating the same user.

Procedures are provided in the following sections:

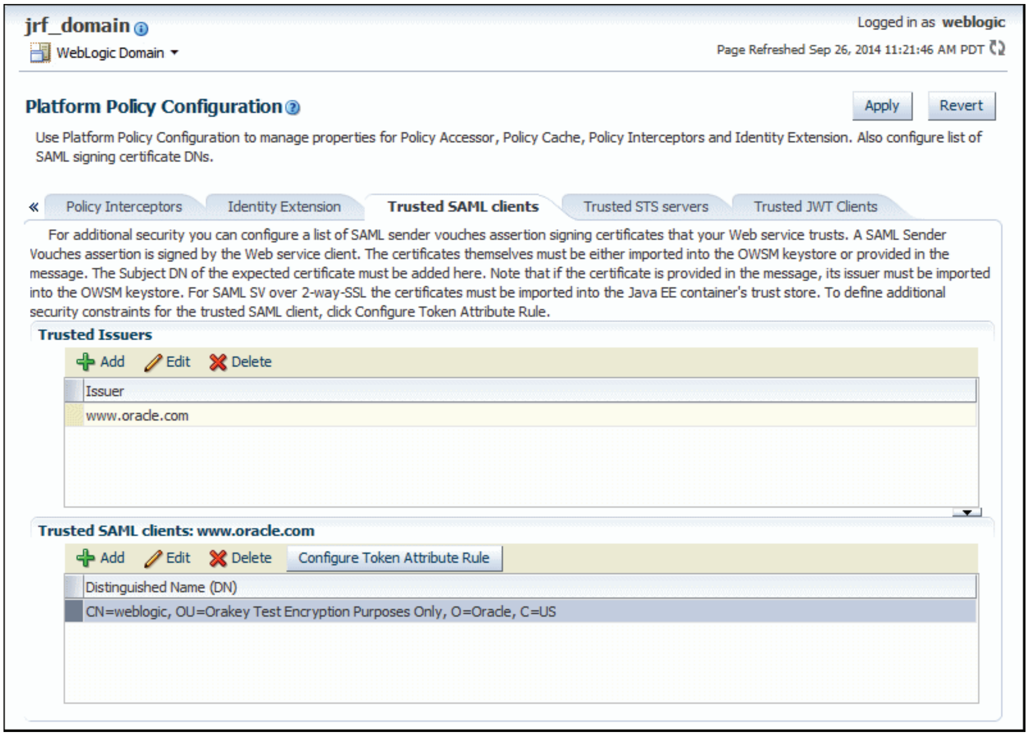
- Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control
- Configuring Token Attribute Rules for Trusted Issuers Using WLST
- Deleting a Token Attribute Rule Using WLST

14.12.3.1 Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control

To configure token attribute rules using Fusion Middleware Control, follow these steps:

1. Define a DN list. For information on defining a DN list in Fusion Middleware Control, see "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23.
2. To define a token attribute rule, select the **Trusted SAML clients**, **Trusted STS servers**, or **Trusted JWT clients** tab.
3. Select an **Issuer** in the top panel, then select a DN in the bottom panel, as illustrated in Figure 14–13.

Figure 14–13 Trusted Issuer and Trusted SAML Clients



Use Platform Policy Configuration to manage properties for Policy Accessor, Policy Cache, Policy Interceptors and Identity Extension. Also configure list of SAML signing certificate DNs.

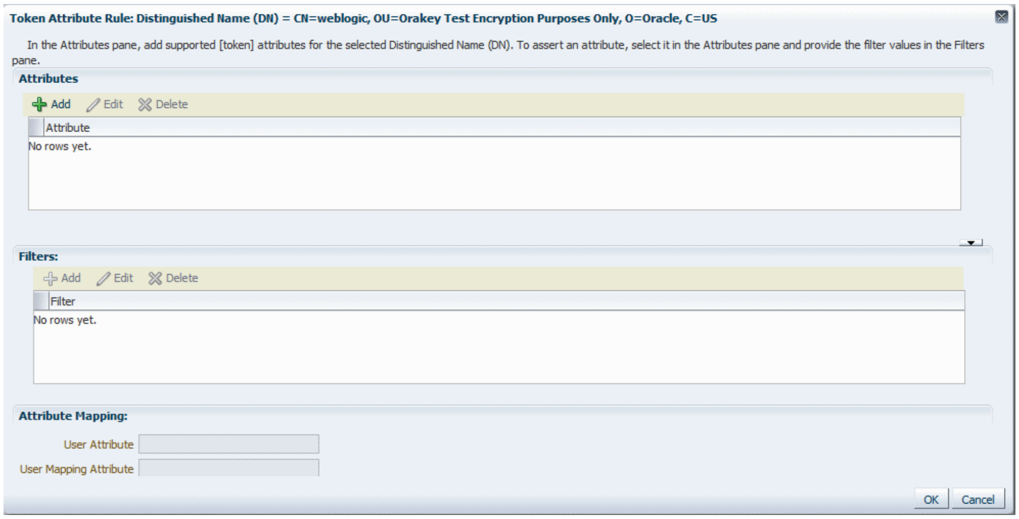
For additional security you can configure a list of SAML sender vouches assertion signing certificates that your Web service trusts. A SAML Sender Vouches assertion is signed by the Web service client. The certificates themselves must be either imported into the OWSM keystore or provided in the message. The Subject DN of the expected certificate must be added here. Note that if the certificate is provided in the message, its issuer must be imported into the OWSM keystore. For SAML SV over 2-way-SSL the certificates must be imported into the Java EE container's trust store. To define additional security constraints for the trusted SAML client, click Configure Token Attribute Rule.

Issuer
www.oracle.com

Distinguished Name (DN)
CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US

4. Click **Configure Token Attribute Rule** to open the Token Attribute Rule window, as shown in Figure 14–14.

Figure 14–14 Token Attribute Rule Page



Token Attribute Rule: Distinguished Name (DN) = CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US

In the Attributes pane, add supported [token] attributes for the selected Distinguished Name (DN). To assert an attribute, select it in the Attributes pane and provide the filter values in the Filters pane.

Attributes

Attribute
No rows yet.

Filters:

Filter
No rows yet.

Attribute Mapping:

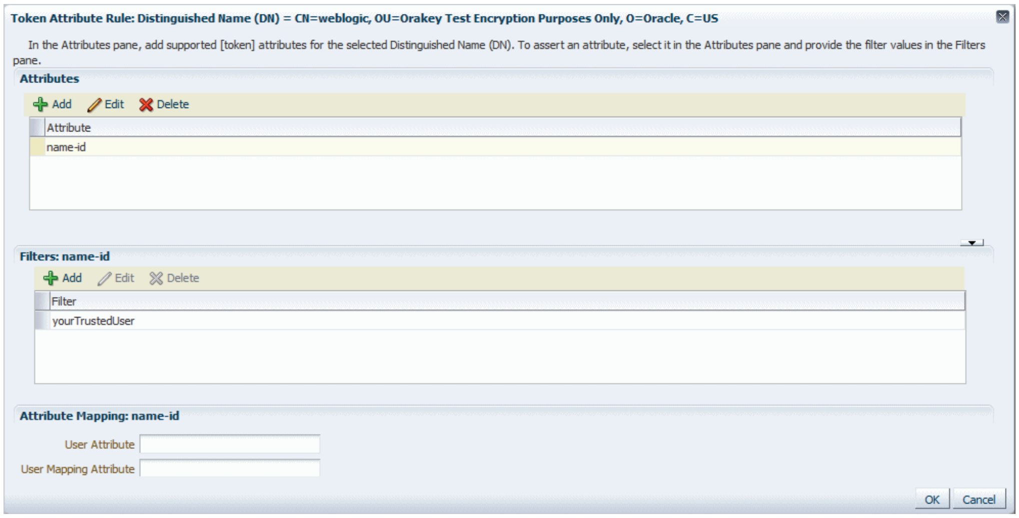
User Attribute

User Mapping Attribute

5. In the Attribute pane, click **Add** to add an attribute for the DN.
6. In the Token Attribute Rule: Add Attribute popup, enter `name-id` to assert a subject name ID and click **OK**.
7. Select the attribute and, in the Filters pane, click **Add** to enter a filter value in the Token Attribute Rule: Add Filter Value popup and click **OK**.

   The value that you add in the **Filter** field can be a complete name, such as `yourTrustedUser` as illustrated in Figure 14–15. You can also enter a name pattern with a wildcard character (`*`), such as `yourTrusted*`.

Figure 14–15 A Token Attribute Page Populated with a Filter



Token Attribute Rule: Distinguished Name (DN) = CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US

In the Attributes pane, add supported [token] attributes for the selected Distinguished Name (DN). To assert an attribute, select it in the Attributes pane and provide the filter values in the Filters pane.

Attributes

Attribute
name-id

Filters: name-id

Filter
yourTrustedUser

Attribute Mapping: name-id

User Attribute

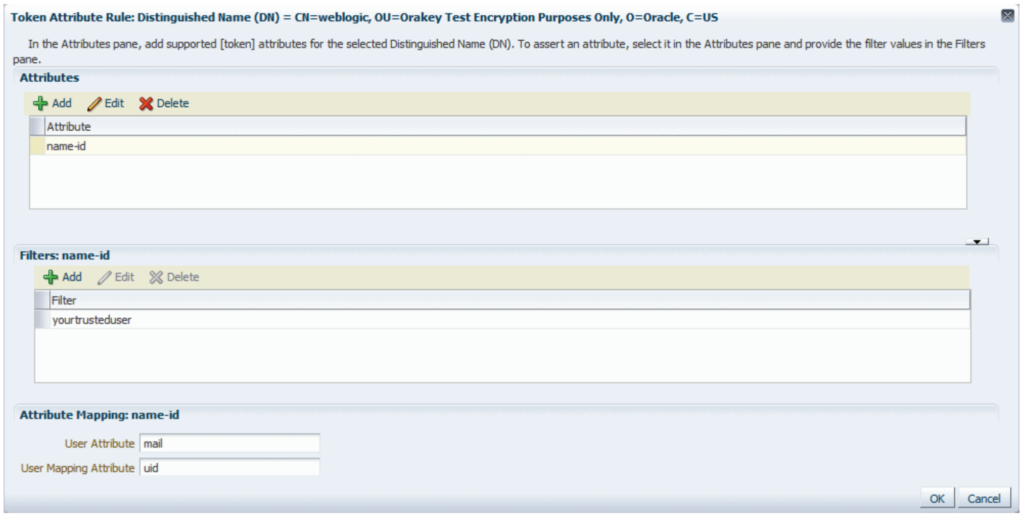
User Mapping Attribute

8. Repeat the previous step to add additional filters.
9. Optionally, you can use the Attribute Mapping fields for the `name-id` attribute to map the local user attribute for the subject name ID to the local user attributes to authenticate the trusted user.

Note: To use the user-attribute mapping feature when the subject is defined in a SAML policy, you must set `subject.precedence` to true in the attached policy.

- User Attribute – The local name of the user attribute in the local identity store that the subject name ID corresponds to (for example, `mail`).
- User Mapping Attribute – The value of the local name of the user attribute in the local identity store that the subject name ID maps to for authentication (for example, `uid`).

Figure 14–16 A Token Attribute Page Populated with a Filter and Mapping



Token Attribute Rule: Distinguished Name (DN) = CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US

In the Attributes pane, add supported [token] attributes for the selected Distinguished Name (DN). To assert an attribute, select it in the Attributes pane and provide the filter values in the Filters pane.

Attributes

Attribute
name-id

Filters: name-id

Filter
yourtrusteduser

Attribute Mapping: name-id

User Attribute: mail

User Mapping Attribute: uid

10. When you have finished defining the attribute filters, click **OK** to close the Token Attribute Rule window.

14.12.3.2 Configuring Token Attribute Rules for Trusted Issuers Using WLST

You can add additional security constraints by using the `setWSMTokenIssuerTrustAttributeFilter` and `setTokenIssuerTrustAttributeMapping` WLST commands to define token attribute rules for a trusted DN. The attribute rules can be applied to a subject name ID.

Note: You must first use the `setWSMTokenIssuerTrust` command to configure a list of trusted DN names for an issuer. See "Configuring an Issuer and its DN List Using WLST" on page 14-25.

The name ID and the attribute can contain a filter with multiple value patterns with the `setWSMTokenIssuerTrustAttributeFilter` command:

```
setWSMTokenIssuerTrustAttributeFilter(dn, attr-name, [filters])
```

The user attribute can be mapped to another user ID with the `setTokenIssuerTrustAttributeMapping` command:

```
setTokenIssuerTrustAttributeMapping('DN', 'attr-name', 'mapping',
'user-attribute', 'user-mapping-attribute')
```

Use Case Examples

In the following example, the name ID `yourTrustedUser` is set as a trusted user for the `weblogic` trusted DN:

```
setWSMTokenIssuerTrustAttributeFilter('CN=weblogic, OU=Orakey Test Encryption
Purposes Only,
O=Oracle, C=US','name-id', ['yourTrustedUser'])

Starting Operation setWSMTokenIssuerTrustAttributeFilter ...
The token attribute filter are successfully set
```

In the following example, the subject name ID `jdoe` is added to the list of trusted users for the `weblogic` trusted DN:

```
setWSMTokenIssuerTrustAttributeFilter('CN=weblogic, OU=Orakey Test Encryption
Purposes Only,
O=Oracle, C=US','name-id', ['yourTrustedUser', 'jdoe'])

Starting Operation setWSMTokenIssuerTrustAttributeFilter ...
The token attribute filter are successfully set
```

In the following example, the `mail` attribute for the Subject ID in the token is mapped to the `uid` attribute.

```
setTokenIssuerTrustAttributeMapping('CN=weblogic, OU=Orakey Test Encryption
Purposes Only,O=Oracle, C=US','name-id','mail','uid')

Starting Operation setWSMTokenIssuerTrustAttributeMapping ...
The token attribute mapping are successfully set
```

For more information about these WLST commands, see "Web Services Custom WLST Commands" in the *WebLogic Scripting Tool Command Reference*.

14.12.3.3 Deleting a Token Attribute Rule Using WLST

You can delete a token attribute rule associated with a DN list by using the `deleteWSMTokenIssuerTrustAttributeRule` WLST command:

```
deleteWSMTokenIssuerTrustAttributeRule(dn)
```

In the following example, the token attribute rule associated with the `weblogic` trusted DN is deleted.

```
deleteWSMTokenIssuerTrustAttributeRule('CN=weblogic, OU=Orakey Test Encryption
Purposes Only, O=Oracle, C=US')
```

For more information about these WLST commands, see "Web Services Custom WLST Commands" in the *WebLogic Scripting Tool Command Reference*.

14.13 Using a Token Attribute Rule for Client Identity Mapping

You can create a token attribute rule to support client-side identity/attribute mapping for user-identity propagation with SAML/JWT tokens.

Consider the cross-identity-domain use case, in which a common user attribute such as email might be more conveniently used as the subject in the token. This requires mapping the authenticated Subject principal to a user attribute to create tokens. To do this, you create a token attribute rule to specify and manage the mappings for various targets.

User identity mapping is supported for identity propagation for end users for all the client policies with SAML or JWT tokens for SOAP or REST when the following is true:

- The authenticated user Subject is available.
- The property `subject.precedence=true`.

You use the `setWSMTokenIssuerTrustAttributeMapping` WLST command to configure client identity mapping. For more information about this WLST command see "Web Services Custom WLST Commands" in the *WebLogic Scripting Tool Command Reference*.

```
setWSMTokenIssuerTrustAttributeMapping('http://oracle.com',  'name-id', '',
'mail')
```

where:

- *`http://oracle.com`* is the identifier of the rule, which is a URL pattern, for the token attribute rule for which modifications are done.
- *`name-id`* indicates that the mapping is applied to the Subject `name id` for the out-going SAML/JWT token.
- Blank. Optional. The name of the local user attribute the value of the attribute corresponds to.
- *`mail`* is the name of the local user attribute to map to.

14.13.1 Mapping Rules

The identifier is a URL pattern. Given a service URL, a token attribute rule applies to it if its identifier is part of the service URL.

```
http(s)://host; http(s)://host/<root path>
```

The mapping rule applies to all web service invocations for which the target service URL has the identifier as the root.

URL *`u1`* is considered part of a URL *`u2`* if:

- The protocol for *`u1`* and *`u2`* match.
- The host for *`u1`* and *`u2`* match.
- If the path in *`u1`* is available, then the path in *`u2`* starts with the path in *`u1`*.

If multiple rules (rule-1, rule-2, …, rule-n) apply to a service URL, then the rule whose identifier has the identifiers of all the other rules takes precedence among the rules.

Mapping Rules Example

Assume you have the following two rules:

```
 Rule 1: setWSMTokenIssuerTrustAttributeMapping('http://oracle.com', 'name-id', '
', 'mail')

Rule2: setWSMTokenIssuerTrustAttributeMapping('http://oracle.com/root1',
'name-id', ' ', 'uid')
```

For a web service call with a service URL = `http://oracle.com/root1/service1`, both *`Rule 1`* and *`Rule 2`* apply to the service. However, *`Rule 2`* takes precedence and is enforced at runtime.

14.14 Configuring Oracle WSM with a Domain-Wide Administration Port

When your domain is configured to use an administration port, all tasks performed by administrators must go through this port. By default, the Oracle WSM Policy Manager is targeted to a Managed Server. To use the Policy Manager with an administration port, you must target the Policy Manager to the Managed Server and the Administration Server.

For more information about the administration port, refer to the following topics:

- "Understanding Network Channels" in *Configuring Server Environments for Oracle WebLogic Server*.
- "Configure the domain-wide administration port" in *Oracle WebLogic Server Administration Console Help*.

Configuring Oracle WSM with a domain-wide administration port requires two main steps:

Step 1. Use the WebLogic Administration Console to target the Policy Manager to the Administration Server.

1. Access the WebLogic Administration Console as described in "Accessing Oracle WebLogic Administration Console" on page 1-6.

   You can also access the WebLogic Administration Console from the WebLogic Domain Home page in Fusion Middleware Control as follows:

   a. Log in to Fusion Middleware Control as described in "Accessing Oracle Enterprise Manager Fusion Middleware Control" on page 1-5.

   b. In the navigator pane, expand **WebLogic Domain** and select the domain for which you want to access the Administration Console.

      The WebLogic Domain home page is displayed.

   c. From the **WebLogic Domain** menu, select **WebLogic Server Administration Console**. Alternatively, you can click the **Oracle WebLogic Server Administration Console** link in the Summary section of the page.

   d. In the Welcome page, log in using a valid username and password.

2. In the left pane of the Console, click **Deployments**. A table in the right pane displays all deployed Enterprise Applications and Application Modules.

3. In the table, locate the wsm-pm application you want to re-target and click on its name. You may have to click **Next** several times to find the application.

   The Settings page for the application is displayed.

4. Click the **Targets** tab.

   The servers to which the Policy Manager application is targeted are shown in the Current Targets column of the table.

5. To target the wsm-pm application to the AdminServer, click **Change Targets**.

6. In the Servers box, select **AdminServer** if it is not already selected and click **Yes**.

   The changes are activated automatically.

Step 2. Use Fusion Middleware Control to specify the policy accessor URL for the Policy Manager on the Administration Server.

1. Access the Platform Policy Configuration page, as described in "Configuring Platform Policy Properties" on page 14-15.

2. Select the **Policy Accessor** tab.
3. Click **Add** to specify the JNDI provider URL for the Administration Server.

   In the Add Property window, specify the following values:

   a. In the **Name** field, enter the JNDI provider URL property as `java.naming.provider.url`.

   b. In the **Value** field, enter the JNDI provider's URL for the Administration Server, such as `t3s://`*`host`*`:`*`admin_port`*`/wsm-pm`.

      For example, `t3s://localhost:9002/wsm-pm`.

      When an override port is configured for the Administration Server, use:

      `t3s://`*`admin_server`*`:`*`admin_server_override_port`*

      This specifies the location of a Policy Manager running on the Administration Server.

      Note: When using an override port, the Policy Manager Validator page will still be located at `https://`*`admin_server_host`*:*`normal_https_port`*`/wsm-pm`, and not on the override port. For more information, see "Diagnosing Problems with Oracle WSM Policy Manager" on page 16-1.

   c. Click **OK**.

14.15 Setting Up the Java Object Cache

To protect against replay attacks, the `wss_username_token_client_policy` and `wss_username_token_service_policy` policies provide the option to require a nonce in the username token. A *nonce* is a unique number that can be used only once in a SOAP request and is used to prevent replay attacks.

The nonce is cached to prevent its reuse. However, in a cluster environment you must take steps to synchronize this cache across the Managed Servers. Otherwise, a request sent to a Web service running on one server can be replayed and sent to another Managed Server, where it will be processed. Oracle WSM uses an instance of Java Object Cache (JOC) to cache the nonce.

You use the *ORACLE_HOME/bin/configure-joc.py* Python script to configure the JOC on all of the Managed Servers in distributed mode. The script runs in WLST online mode and expects the Administration Server to be up and running.

> **Note:** After configuring the Java Object Cache, restart all affected Managed Servers for the configurations to take effect.

14.15.1 Running the configure-joc.py Script

To enable the JOC in distributed mode, perform the following steps:

1. Connect to the Administration Server using the command-line Oracle WebLogic Scripting Tool (WLST), for example:

```
ORACLE_HOME/oracle_common/common/bin/wlst.sh
$ connect()
```

   Enter the Oracle WebLogic Administration user name and password when prompted.

2. After connecting to the Administration Server using WLST, start the script using the *execfile* command, for example:

   ```
   wls:/mydomain/serverConfig>execfile('ORACLE_HOME/bin/configure-joc.py')
   ```

3. Configure JOC for all the Managed Servers for a given cluster.

   Enter 'y' when the script prompts whether you want to specify a cluster name, and also specify the cluster name and discover port, when prompted. This discovers all the Managed Servers for the given cluster and configures the JOC for each Managed Server. The discover port is common for the entire JOC configuration across the cluster. For example:

   ```
   Do you want to specify a cluster name (y/n) <y>
   Enter Cluster Name : Spaces_Cluster
   Enter Discover Port : 9988
   ```

   Here is a walkthrough for using *configure-joc.py* for HA environments:

   ```
   execfile('ORACLE_HOME/bin/configure-joc.py')
   .
   Enter Hostnames (eg host1,host2) : SOAHOST1, SOAHOST2
   .
   Do you want to specify a cluster name (y/n) <y>y
   .
   Enter Cluster Name : Spaces_Cluster
   .
   Enter Discover Port : 9988
   .
   Enter Distribute Mode (true|false) <true> : true
   .
   Do you want to exclude any server(s) from JOC configuration (y/n) <n> n
   ```

The script can also be used to perform the following JOC configurations:

- Configure JOC for all specified Managed Servers.

  Enter 'n' when the script prompts whether you want to specify a cluster name, and also specify the Managed Server and discover port, when prompted. For example:

  ```
  Do you want to specify a cluster name (y/n) <y>n
  Enter Managed Server and Discover Port (eg WLS_Spaces1:9988, WLS_Spaces2:9988)
  : WLS_Spaces1:9988,WLS_Spaces2:9988
  ```

- Exclude JOC configuration for some Managed Servers.

  The script allows you to specify the list of Managed Servers for which the JOC configuration "DistributeMode" will be set to 'false'. Enter 'y' when the script prompts whether you want to exclude any servers from JOC configuration, and enter the Managed Server names to be excluded, when prompted. For example:

  ```
  Do you want to exclude any server(s) from JOC configuration (y/n) <n>y
  Exclude Managed Server List (eg Server1,Server2) : WLS_Spaces1,WLS_Spaces3
  ```

- Disable the distribution mode for all Managed Servers.

  The script allows you to disable the distribution to all the Managed Servers for a specified cluster. Specify 'false' when the script prompts for the distribution mode. By default, the distribution mode is set to 'true'.

Verify JOC configuration using the CacheWatcher utility. For more information, see "Running CacheWatcher" in *Oracle Fusion Middleware High Availability Guide*.

You can configure the Java Object Cache (JOC) using the HA Power Tools tab in the Oracle WebLogic Administration Console as described in "Using HA Power Tools" in the *Oracle Fusion Middleware High Availability Guide*.

14.16 Modifying the Default User

The Oracle WSM Agent run time uses the OracleSystemUser account by default to communicate to the server.

To configure a different default user, perform the following steps:

- Configure an Authentication Provider
- Configure the Credential Store Provider
- Configure the Platform Policy Configuration
- Modify the User's Group or Role

Configure an Authentication Provider

To configure an authentication provider, perform the following steps:

1. Configure an authentication provider, as described in "Configure Authentication and Identity Assertion providers" in *Oracle WebLogic Server Administration Console Help*.

   - Select the name of the realm you are configuring (for example, myrealm).
   - In the Create a New Authentication Provider page, enter the name for Authentication Provider (for example, OID) and select the type Oracle Internet Directory Authenticator.
   - In the Settings section, set Control Flag to **OPTIONAL**.

   In the Provider Specific tab, enter the following:

   - Host: the LDAP provider URL
   - Port: port number
   - Principal: administrator user details (the new default user)

     For example, CN=orcladmin,CN=Users,DC=us,DC=oracle,DC=com
   - Credential: password for LDAP
   - Confirm Credential: password for LDAP
   - User Base DN

     For example, CN=Users,DC=us,DC=oracle,DC=com
   - Group Base DN

     For example, CN=Groups,DC=us,DC=oracle,DC=com

2. Restart WebLogic Server.

Configure the Credential Store Provider

Configure the credential store provider as described in "Adding Keys and User Credentials to the Credential Store" on page 10-19 with the following parameters:

- If a map does not already exist, select Create Map and enter the map name `oracle.wsm.security`.
- In the Credential Store Provider table, select `oracle.wsm.security`.

- In the Create Key dialog, enter the appropriate key; for example, `OID`. Enter the user name and password of the new default user (in this example, `orcladmin` and *`password`*).

Configure the Platform Policy Configuration

To configure the Platform Policy, perform the following steps:

1. Log into Fusion Middleware Control with the new default user account.
2. In the navigator pane, expand WebLogic Domain to view the domains.
3. Select the domain for which you want to manage properties.
4. Select **WebLogic Domain** > **Web Services** > **Platform Policy Configuration**.

   The Platform Policy Configuration page appears.
5. Select the **Policy Accessor** tab.
6. Click **Add** in the Policy Access Properties section.
7. In the Add New Configure Property dialog, enter the following:
   - Enter the name jndi.lookup.csf.key. This property provides credential configuration (java.naming.security.principal and java.naming.security.credentials) and is used when an account in the LDAP directory is configured to connect with the Oracle WSM Policy Manager.
   - Enter the value (in this example, OID).

   > **Note:** The csf-key that you specify in this step must match the csf-key specified for the Policy Manager administrative user in the credential store. For more information, see "Configure the Credential Store Provider".
8. Click **OK**.
9. Click **Apply** and restart WebLogic Server.

Modify the User's Group or Role

Oracle WSM Policy Manager uses the logical role (policy.Accessor) to secure EJBs that are accessed by the Oracle WSM Agent runtime to access the policies. By default, the policy.Accessor role is mapped to the groups OracleSystemGroup and Administrators. Oracle WSM Agent run time uses the OracleSystemUser identity to access wsm-pm. The new default user must either be included in the Administrator or OracleSystemGroup (if the groups exist), or be mapped to the logical role policy.Accessor (if the Administrator or OracleSystemGroup groups do not exist).

To ensure the user has the required role, perform the following steps:

1. If the Administrator or OracleSystemGroup groups exist in the LDAP or identity store, perform the following:

   a. In LDAP, add the user that you would like to use as a default administrative user.

   b. In WebLogic Server Administration Console, ensure that the user exists in the Administrator group. For more information, see "Manage Users and Groups" in *Oracle WebLogic Server Administration Console Help*.

2. If the Administrator or OracleSystemGroup groups do not exist in the LDAP or identity store, map the new user to the required logical role and redeploy the wsm-pm application using the modified deployment plan. To map the new user or existing users (belonging to a group other than Administrator or OracleSystemGroup), perform the following steps:

   a. Create a deployment plan for deploying wsm_pm.ear. Example 14–4 describes a sample deployment plan. A sample deployment plan, shipped with WebLogic, is available in the *ORACLE_HOME/modules/oracle.wsm.pm_11.1.1/prov* folder. Modify the section `@to_be_replaced@` with the new user.

Example 14–4 Sample Deployment Plan

```
<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan xmlns="http://xmlns.oracle.com/weblogic/deployment-plan"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.oracle.com/weblogic/deployment-plan
 http://xmlns.oracle.com/weblogic/deployment-plan/1.0/deployment-plan.xsd">
  <application-name>oracle.wsm.pm_11.1.1</application-name>
  <variable-definition>
    <variable>
      <name>SecurityRoleAssignment_ejbRole_PrincipalName</name>
      <value>@to_be_replaced@</value>
    </variable>
  </variable-definition>
  <module-override>
    <module-name>wsm-pmserver-wls.jar</module-name>
    <module-type>ejb</module-type>
    <module-descriptor external="false">
      <root-element>weblogic-ejb-jar</root-element>
      <uri>META-INF/weblogic-ejb-jar.xml</uri>
      <variable-assignment>
             <name>SecurityRoleAssignment_ejbRole_PrincipalName</name>

<xpath>/weblogic-ejb-jar/security-role-assignment/[role-name="policy.Accessor"]
/principal-name</xpath>
             <operation>replace</operation>
    </variable-assignment>
    </module-descriptor>
  </module-override>
</deployment-plan>
```

   b. Redeploy the EAR.

      For more information, see "Deploying an Application with a Deployment Plan" in *Deploying Applications to Oracle WebLogic Server*.

14.17 Changing the JMS System User for Asynchronous Web Services

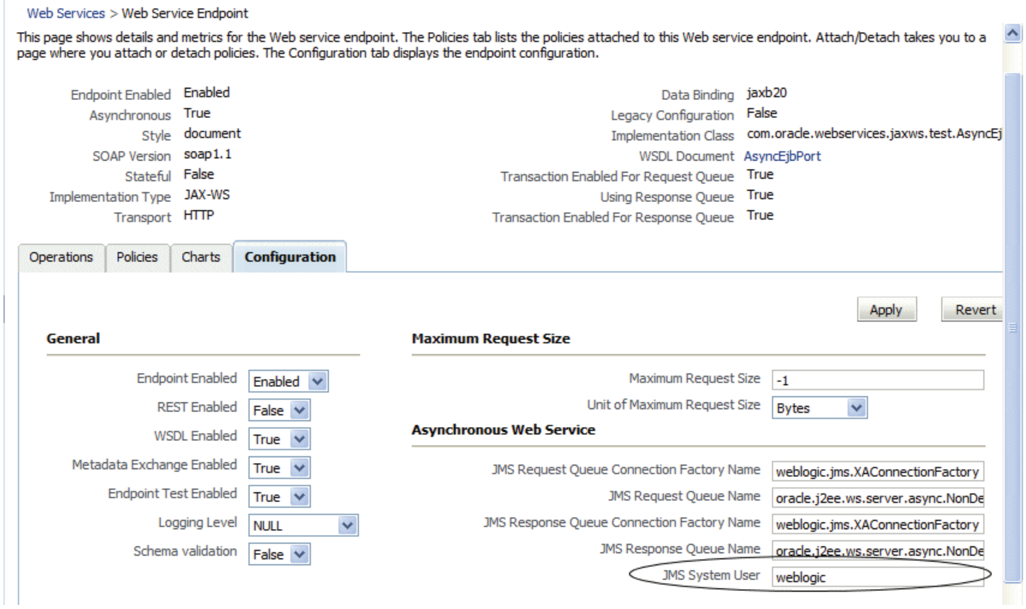
By default, the JMS System User is set as the OracleSystemUser. For most users, this default value is sufficient. However, if you must change this value to a custom user in your security realm, you can do so by changing the value of the user in Oracle Enterprise Manager Fusion Middleware Control and in the WebLogic Server Administration Console as described in the following procedure.

To change the JMS System User:

1. Access the **Configuration** tab on the Web Service Endpoint page for the asynchronous Web service as described in "Configuring Asynchronous Web Services" on page 6-26.
2. Enter the name of the custom user in the **JMS System User** field and click **Apply**. See Figure 14–17.

> **Note:** The custom user must exist in the security realm and have the permissions required to access the JMS resources.

Figure 14–17 Setting the JMS System User for Asynchronous Web Services



Web Services > Web Service Endpoint

This page shows details and metrics for the Web service endpoint. The Policies tab lists the policies attached to this Web service endpoint. Attach/Detach takes you to a page where you attach or detach policies. The Configuration tab displays the endpoint configuration.

Endpoint Enabled	Enabled	Data Binding	jaxb20
Asynchronous	True	Legacy Configuration	False
Style	document	Implementation Class	com.oracle.webservices.jaxws.test.AsyncEj
SOAP Version	soap1.1	WSDL Document	AsyncEjbPort
Stateful	False	Transaction Enabled For Request Queue	True
Implementation Type	JAX-WS	Using Response Queue	True
Transport	HTTP	Transaction Enabled For Response Queue	True

General

Endpoint Enabled	Enabled
REST Enabled	False
WSDL Enabled	True
Metadata Exchange Enabled	True
Endpoint Test Enabled	True
Logging Level	NULL
Schema validation	False

Maximum Request Size

Maximum Request Size	-1
Unit of Maximum Request Size	Bytes

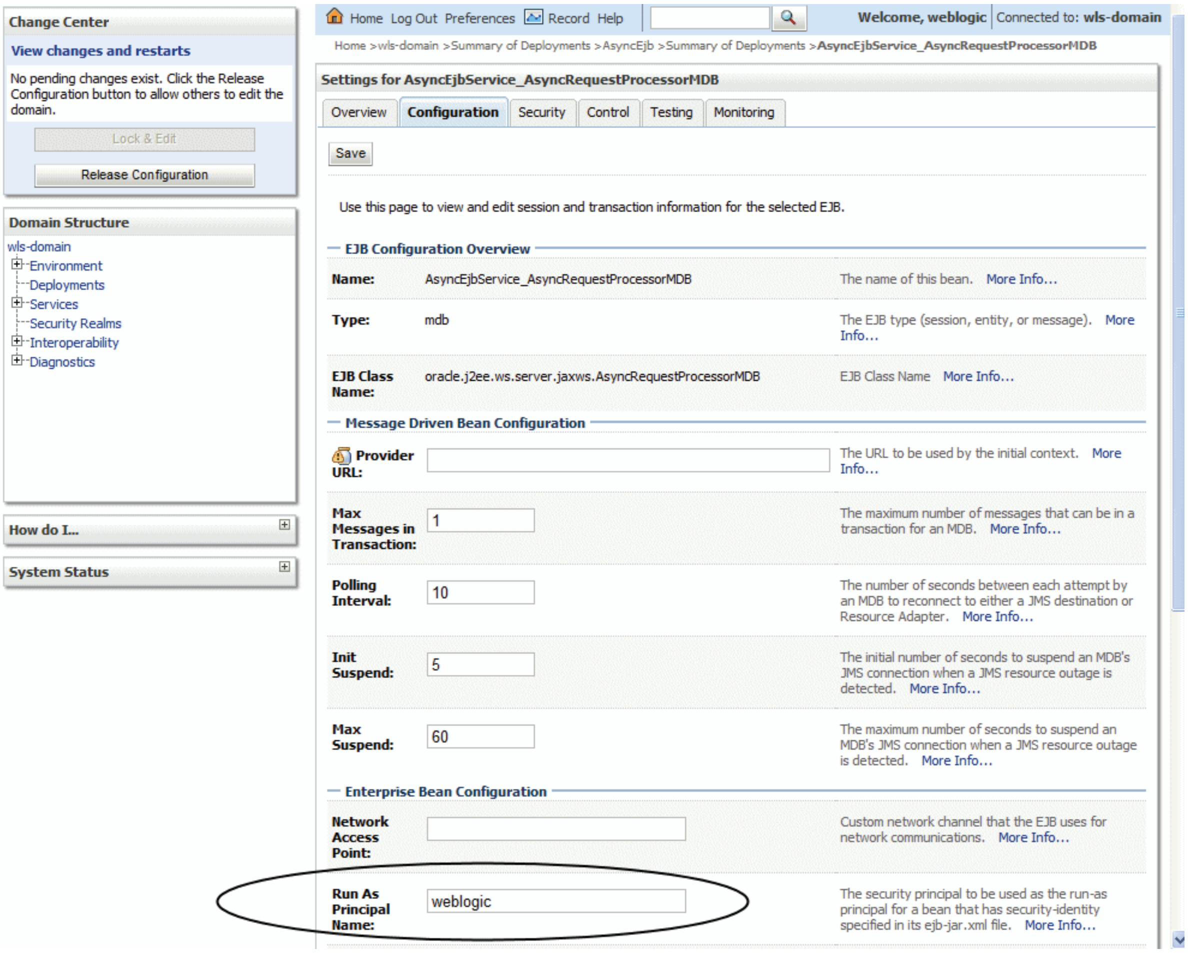
Asynchronous Web Service

JMS Request Queue Connection Factory Name	weblogic.jms.XAConnectionFactory
JMS Request Queue Name	oracle.j2ee.ws.server.async.NonDe
JMS Response Queue Connection Factory Name	weblogic.jms.XAConnectionFactory
JMS Response Queue Name	oracle.j2ee.ws.server.async.NonDe
JMS System User	weblogic

3. Access the WebLogic Server Administration Console. To do so from Fusion Middleware Control, select the domain in the navigator pane. From the **WebLogic Domain** menu, select **WebLogic Server Administration Console**.
4. Log into the WebLogic Server Administration Console using a valid username and password with the required administrative privileges.
5. Click **Deployments** in the Domain Structure pane and navigate to the corresponding *`service`*`_AsynchRequestProcessorMDB` or *`service`*`_AsynchResponseProcessorMDB` MDBs. In these MDB names, *`service`* is the name of the asynchronous service for which you are changing the user name.
6. In the Change Center, select **Lock & Edit**.
7. Select the MDB name for the request or response MDB. (You must update the user name for both the request and response MDBs.) In the Settings page, select the **Configuration** tab.
8. In the Enterprise Bean Configuration section of the page, enter the custom user name in the **Run As Principal Name** field and click **Save**. See Figure 14–18.

   Note that the user name you enter in this field must match the user name you entered for the JMS System User in Fusion Middleware Control.

Figure 14–18 WebLogic Server Administration Console Update for JMS System User



No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

Settings for AsyncEjbService_AsyncRequestProcessorMDB

Use this page to view and edit session and transaction information for the selected EJB.

EJB Configuration Overview

Name:	AsyncEjbService_AsyncRequestProcessorMDB	The name of this bean. More Info...
Type:	mdb	The EJB type (session, entity, or message). More Info...
EJB Class Name:	oracle.j2ee.ws.server.jaxws.AsyncRequestProcessorMDB	EJB Class Name More Info...

Message Driven Bean Configuration

Provider URL:		The URL to be used by the initial context. More Info...
Max Messages in Transaction:	1	The maximum number of messages that can be in a transaction for an MDB. More Info...
Polling Interval:	10	The number of seconds between each attempt by an MDB to reconnect to either a JMS destination or Resource Adapter. More Info...
Init Suspend:	5	The initial number of seconds to suspend an MDB's JMS connection when a JMS resource outage is detected. More Info...
Max Suspend:	60	The maximum number of seconds to suspend an MDB's JMS connection when a JMS resource outage is detected. More Info...

Enterprise Bean Configuration

Network Access Point:		Custom network channel that the EJB uses for network communications. More Info...
Run As Principal Name:	weblogic	The security principal to be used as the run-as principal for a bean that has security-identity specified in its ejb-jar.xml file. More Info...

The configuration changes must be saved in a new deployment plan.

9. Use the Save Deployment Plan Assistant to save the new deployment plan.
10. Repeat steps 7 and 8 for the second MDB. The changes are automatically saved to the new deployment plan.
11. In the Change Center, click **Activate Changes**.
12. Redeploy the application. For more information, see Chapter 5, "Deploying Web Services Applications."

15 Managing Application Migration Between Environments

This chapter describes how to migrate Web service applications between environments, such as from development or test to production. It includes the following sections:

15.1 Overview of Web Service Application Migration

To migrate Web service applications independently between environments, such as from test to production, or in a scaled clustered environment, you must export the policies and the deployment configuration information to the new environment so that you can deploy the application. Depending on your configuration, you may also need to migrate policy configuration artifacts and policy assertion templates.

A deployment descriptor is an XML file that contains the basic deployment configuration for an application. For WebLogic Server and WebLogic Java EE Web services applications, you create a deployment plan that contains the necessary deployment descriptors for deploying the application in a new environment.

For ADF Business Components and WebCenter services, however, run-time policy changes are persisted in proprietary deployment descriptor (PDD) files: oracle-webservices.xml and oracle-webservices-client.xml. Because these files are not included in the WebLogic deployment plan or exported with any other deployment descriptors, you must export and import these PDD files separately. You must also export and import these PDD files separately if you are scaling your application in a clustered environment.

Note that the following Oracle Infrastructure Web services components provide different configuration management mechanisms.

- For a SOA composite, Web services and Oracle WSM configurations are persisted in a composite.xml file which is included in a configuration plan used for deployment configuration. The SOA framework provides its own mechanism for composite services and configuration lifecycle and synchronizations.

- ADF Web Service data control configuration stores connection details for WebCenter services in a connections.xml file and all post-deployment changes as customizations in the Metadata Services (MDS) repository.

The general steps for migrating a Web service application from a development or test environment to a production environment are as follows:

1. Install and configure the production environment with the components that you need.
2. Migrate security information, such as users and groups, the identity and policy stores, and credentials. For more information, see "Migrating Policy Configuration" on page 15-5.
3. Migrate policies and deployment configuration data as required. For more information, see "Migrating Policies" on page 15-4 and "Migrating Deployment Descriptors" on page 15-7. Modify any information that is specific to the new environment such as host name or ports.
4. Deploy the applications in the new environment.

For information about migrating Fusion Middleware applications between environments, see "Advanced Administration: Expanding Your Environment" in *Oracle Fusion Middleware Administrator's Guide*.

15.2 Overview of Horizontal Policy Migration

The following steps describe a typical scenario for how to create a policy and migrate the policy horizontally through the different stages of the application development and deployment cycles.

1. Use Oracle Enterprise Manager Fusion Middleware Control to create a policy.

   For more information, see "Creating Web Service Policies" on page 7-4.

2. Export the policy to a file.

   For more information, see "Migrating Policies" on page 15-4.

3. Copy the policy file to policy store location in the Oracle JDeveloper environment.
4. Create a Web service in Oracle JDeveloper and attach the policy to the Web service.

   For more information, see "Using Policies with Web Services" in the "Developing with Web Services" section of the JDeveloper online help.

5. Deploy the Web service to the staging server, and test the Web service.

   For more information, see "Developing Web Services" in the JDeveloper online help.

6. Import the policy to the production server environment.

   For more information, see "Migrating Policies" on page 15-4.

7. Migrate the following information, as required:

   - Policy configuration. See "Migrating Policy Configuration" on page 15-5.
   - Assertion templates. See "Migrating Assertion Templates" on page 15-7.

8. Deploy the application into the production environment, and test the Web service.

See "Deploying Web Services Applications" on page 5-1 and "Testing Web Services" on page 12-1.

15.3 Sample Use Cases for Deployment Descriptor Migration

The following sections provide sample use cases for ADF Business Control or WebCenter Web Service applications for which you must migrate the PDD files.

15.3.1 Scaling a Deployed ADF Business Control or WebCenter Web Service Application in a Cluster

The following steps describe a sample use case for scaling a deployed ADF Business Control or WebCenter Web service application within a cluster.

1. Deploy an ADF Business Control or WebCenter Web service application in a clustered WebLogic Server domain consisting of, for example, an Administration Server and two Managed Servers (MServer1 and MServer2). For deployment information, see Chapter 5, "Deploying Web Services Applications."
2. Using Fusion Middleware Control or WLST, modify the policy configuration. For example, attach a policy named `oracle/wss_username_token_service_policy` to the Web service on MServer2. For more information, see Chapter 8, "Attaching Policies to Web Services."

   The configuration changes are persisted in the PDD files.
3. Restart the application.
4. Export the PDD to a JAR file using the `exportJRFWSApplicationPDD` command. For more information, see "Migrating Deployment Descriptors" on page 15-7.
5. Using the WebLogic Server Administration Console, clone a new Managed Server named MServer3 in the cluster from MServer2. For more information, see "Clone Servers" in the *Oracle WebLogic Server Administration Console Help*.
6. Start the new Managed Server, MServer3.

   Note that MServer3 does not have the policy `oracle/wss_username_token_service_policy` attached because it was attached after the application was initially deployed.
7. Import the JAR file containing the Oracle WSM PDD files that you created in step 4 to MServer3 using the `importJRFWSApplicationPDD` command. For more information, see "Migrating Deployment Descriptors" on page 15-7.
8. Restart the application.

15.3.2 Propagating Run-time Policy Changes in an ADF Business Control or WebCenter Web Service Environment

The following steps describe a sample use case for propagating run-time policy changes for an ADF Business Control or WebCenter Web service application to all servers in a cluster.

1. Deploy an ADF Business Control or WebCenter Web service application in a clustered WebLogic Server domain consisting of, for example, an Administration Server and three Managed Servers (MServer1, MServer2, and MServer3). For deployment information, see Chapter 5, "Deploying Web Services Applications."

2. Using Fusion Middleware Control or WLST, modify the policy configuration. For example, detach all policies from MServer2. For more information, see Chapter 8, "Attaching Policies to Web Services."

   The configuration changes are persisted in the PDD files.

3. Restart the application.

4. Export the PDD files from MServer2 to a JAR file using the `exportJRFWSApplicationPDD` command. For more information, see "Migrating Deployment Descriptors" on page 15-7.

5. Use the `savePddToAllAppInstancesInDomain` command, with the `restartApp` argument set to `true`. For more information, see "Migrating Deployment Descriptors" on page 15-7

   The policies are now detached from all server instances in the domain and the application is restarted.

15.4 Migrating Policies

You can export individual policies from Oracle Enterprise Manager Fusion Middleware Control. You can then copy the policy to a directory or import the policy to move it to another repository.

> **Note:** If you are using the Oracle RAC database, avoid switching databases when performing a bulk import of policies or metadata upload. This may cause the import to fail.

For details about exporting and importing policies using Fusion Middleware Control, see the following sections in "Managing Web Service Policies" on page 7-1:

- "Exporting Web Service Policies" on page 7-20
- "Importing Web Service Policies" on page 7-7

Alternatively, you can use the `exportRepository` and `importRepository` WLST commands to export and import the policies. The following describes the steps required:

To migrate policies using WLST commands:

1. Export the Oracle WSM policies to a supported archive file, such as a zip file, using the `exportRepository` command.

   For example, to export all Oracle WSM policies to an archive named policies.zip, enter the following:

```
wls:/jrfServer_domain/serverConfig>
exportRepository('/tmp/policies.zip',['policies:oracle/%'])

Exporting "/policies/oracle/binding_authorization_denyall_policy"
Exporting "/policies/oracle/binding_authorization_permitall_policy"
Exporting "/policies/oracle/binding_permission_authorization_policy"
.
.
.
Exporting "/policies/oracle/wss_username_token_over_ssl_service_policy"
Exporting "/policies/oracle/wss_username_token_service_policy"
Successfully exported "84" documents.
```

2. Optionally, you can edit the archive after it has been created. If, for example, you do not want to migrate all the policies to the new environment, you can manually remove them from the archive.
3. Move the archive to the new machine. Ensure that the Oracle WSM Policy Manager is deployed on the new machine.
4. Import the Oracle WSM policies using the `importRepository` command. For example, to import the policies exported in the previous step:

   ```
   wls:/jrfServer_domain/serverConfig> importRepository('/tmp/policies.zip')

   Importing "META-INF/policies/oracle/binding_authorization_denyall_policy"
   Importing "META-INF/policies/oracle/binding_authorization_permitall_policy"
   Importing "META-INF/policies/oracle/binding_permission_authorization_policy"
   .
   .
   .
   Importing "META-INF/policies/oracle/wss_username_token_over_ssl_service_policy"
   Importing "META-INF/policies/oracle/wss_username_token_service_policy"
   Successfully imported "84" documents
   ```

For more information about these WLST commands, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

15.5 Migrating Policy Configuration

The following sections describe how to migrate the configuration artifacts for Oracle WSM policies. Sections include:

- Migrating Keystores
- Migrating Users and Groups
- Migrating Credentials
- Migrating Oracle Platform Security Services Application and System Policies
- Migrating Oracle Platform Security Services Configuration
- Migrating SSL
- Migrating Kerberos Configuration

15.5.1 Migrating Keystores

If you are using message protection policies, you need to migrate your keystores. To migrate keystores:

1. Manually copy your keystores to the new environment.

   For Java SE applications, copy the keystore to a user-defined location. For Java EE applications, copy the keystore to the same directory as the jps-config.xml file, namely *DOMAIN_HOME*/config/fmwconfig.
2. By default, the keystore is named default-keystore.jks. If you have renamed the keystore, you must configure the keystore name in the Oracle Platform Security Services keystore service instance.

For information about configuring the keystore, see "Configuring Keystores for Message Protection" on page 10-9.

15.5.2 Migrating Users and Groups

Users and groups are maintained as part of the WebLogic Server security realm.

To migrate users and groups in embedded LDAP, you can migrate the data using either the Oracle WebLogic Administration Console or WLST. For a complete description of the steps required, see "Migrating Security Data" in *Securing Oracle WebLogic Server*.

To migrate users and groups in an LDAP store, there is no migration path. You need to recreate the users and groups and specify the assignments in the LDAP store in the new environment. See "Configuring Authentication Providers" in *Securing Oracle WebLogic Server*.

15.5.3 Migrating Credentials

There are two types of credentials maintained in the credential store that you may need to migrate:

- Username and password
- Keystore and encryption key passwords

The migration steps are described in the sections below.

15.5.3.1 Migrating Username and Password

If users are stored in an embedded LDAP and migrated, as described in "Migrating Users and Groups" on page 15-6, then you simply migrate the existing credentials to the new credential store. For a complete description of the steps required, see "Migrating Security Data" in *Securing Oracle WebLogic Server*.

If users are stored in an LDAP store, there is no automated migration path. You need to recreate the credentials in the credential store. For more information about configuring credentials, see "Adding Keys and User Credentials to the Credential Store" on page 10-19.

15.5.3.2 Migrating Keystores and Encryption Key Passwords

You can migrate keystores and encryption key passwords manually using the procedure described in "Migrating Credentials Manually" in "Deploying Secure Applications" in *Oracle Fusion Middleware Application Security Guide*.

15.5.4 Migrating Oracle Platform Security Services Application and System Policies

If your Web service uses authorization policies, you must migrate the Oracle Platform Security Services application and system policies that grant permissions. For more information, see "Migrating with the Script migrateSecurityStore" in "Configuring the OPSS Security Store" in *Oracle Fusion Middleware Application Security Guide*.

15.5.5 Migrating Oracle Platform Security Services Configuration

There is no automated migration path for Oracle Platform Security Services configuration. You must recreate the configuration in the new environment.

There are three types of configurations in the Oracle Platform Security Services that you may need to recreate:

- SAML trusted assertion issuer names (applicable for all SAML policies).

If you use the default configuration for SAML trusted issuer configuration, then no migration is required. For information about configuring SAML in the new environment, see "Configuring the SAML and Kerberos Login Modules" on page 10-60.

- Keystore locations and CSF key configuration for keystore and keystore password (applicable for message protection policies only).

  If you use the default configuration for keystores, then no migration is required. For information about configuring keystores in the new environment, see "Configuring Keystores for Message Protection" on page 10-9.

- Keytab location and service principal name (applicable to Kerberos policy).

  For information about configuring the keytab location and service principal name in the new environment, see "Configuring the SAML and Kerberos Login Modules" on page 10-60.

15.5.6 Migrating SSL

There is no automated migration path for SSL configuration. You must configure SSL keystores and settings in the new environment. For more information about configuring SSL keystores and settings in the new environment, see "Configuring Keystores for SSL" on page 10-36.

15.5.7 Migrating Kerberos Configuration

To migrate the Kerberos configuration:

1. Copy the Kerberos configuration file to the new environment, matching the directory structure. The Kerberos configuration file is located in the following locations, based on your operating system:
   - **UNIX**: /etc/krb5.conf
   - **Windows**: C:\windows\krb5.ini
2. Initialize the ticket cache with the correct credentials.

   For more information, see "Using Kerberos Tokens" on page 10-85.

15.6 Migrating Assertion Templates

You can export individual assertion templates from Oracle Enterprise Manager Fusion Middleware Control. You can then copy the policy to a directory or import the policy to move it to another repository.

For details about exporting and importing assertion templates, see the following sections:

- "Exporting an Assertion Template" on page 7-14
- "Importing an Assertion Template" on page 7-15

15.7 Migrating Deployment Descriptors

To deploy an application in a new environment, you must export the application's deployment configuration to the new environment. Refer to the following topics for information about exporting WebLogic Java EE Web services, SOA composites, and ADF data control deployment configuration:

- WebLogic Java EE Web services: "Exporting an Application for Deployment to New Environments" in *Deploying Applications to Oracle WebLogic Server*.
- SOA composites: "Exporting a Running SOA Composite Application" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.
- ADF data controls: "WebCenter Portal Configuration" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

Run-time Web service policy changes to ADF Business Control and WebCenter Web service applications are persisted in proprietary deployment descriptor (PDD) files. To preserve these policy changes in a scaled or new environment, or to propagate these changes in a domain, you must migrate the PDD files using WLST commands. The following procedure describes the steps required to migrate the PDD files.

1. Connect to the running instance of the WebLogic Administration Server in the domain in which the application is deployed. For instructions, see "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Optionally, use the `listWebServices(None,None,true)` command to list all the Web services in all applications and composites in the domain.

   For example:

```
wls:/wls-domain/serverConfig> listWebServices(None,None,true)

/wls-domain/ManagedServer1/j2wbasicPolicy:
    moduleName=j2wbasicPolicy, moduleType=web,
serviceName={http://namespace/}WssUsernameService
    enableTestPage: true
    enableWSDL: true

            JRFWssUsernamePort    http://host:port/j2wbasicPolicy/WssUsername
            enable: true
            enableREST: false
            enableSOAP: true
            maxRequestSize: -1
            loggingLevel: NULL
            security : oracle/wss_username_token_service_policy, enabled=true
        enableWSDL: true

          Attached policy or policies are valid; endpoint is secure.
.
.
.
```

Note: The `listWebServices` command output does not include details on SOA components, including policy attachments.

3. Modify the policy configuration using Fusion Middleware Control or WLST. For example, to attach the policy `oracle/wsmtom_policy` to ManagedServer1 using WLST, enter the following command:

```
wls:/wls-domain/serverConfig> attachWebServicePolicy
('/wls-domain/ManagedServer1/j2wbasicPolicy','j2wbasicPolicy','web',
'WssUsernameService','JRFWssUsernamePort',
'oracle/wsmtom_policy,')
```

The policy change is persisted in the application PDD.

4. Restart the application.

5. Optionally, use the `listWebServices(None,None,true)` command again to verify that the policy is attached to the server instance. For example:

```
wls:/wls-domain/serverConfig> listWebServices(None,None,true)

/wls-domain/ManagedServer1/j2wbasicPolicy :
    moduleName=j2wbasicPolicy, moduleType=web,
serviceName={http://namespace/}WssUsernameService
    enableTestPage: true
    enableWSDL: true

          JRFWssUsernamePort    http://host:port/j2wbasicPolicy/WssUsername
          enable: true
          enableREST: false
          enableSOAP: true
          maxRequestSize: -1
          loggingLevel: NULL
          security : oracle/wss_username_token_service_policy , enabled=true
          mtom : oracle/wsmtom_policy , enabled=true

         Attached policy or policies are valid; endpoint is secure.
```

6. Export the application PDD to a JAR file using the `exportJRFWSApplicationPDD` WLST command.

```
exportJRFWSApplicationPDD(application,pddJarFileName=None)
```

For example, to export the PDD for the j2wbasicPolicy application using the default JAR filename, use the following command:

```
wls:/wls-domain/serverConfig>
exportJRFWSApplicationPDD('/wls-domain/ManagedServer1/j2wbasicPolicy', None)
```

The default name and path to the JAR file is displayed.

```
/tmp/j2wbasicPolicy-PDD-20100115-145338.jar
```

7. If you are scaling your environment, use the WebLogic Server Administration Console to clone a new Managed Server. For more information, see "Clone Servers" in the *Oracle WebLogic Server Administration Console Help*.

If you are migrating your application to a new environment, ensure that the policies and any other configuration artifacts are copied to the new environment. For more information, see "Migrating Policies" on page 15-4 and "Migrating Policy Configuration" on page 15-5.

8. In a scaled environment, start the cloned Managed Server.

Note that the `oracle/wsmtom_policy` is not attached to the cloned Managed Server. To do so using the `listWebServices` command:

```
wls:/wls-domain/serverConfig> listWebServices(None,None,true)
.
.
.
/wls-domain/ClonedManagedServer/j2wbasicPolicy :
    moduleName=j2wbasicPolicy,
moduleType=web,serviceName={http://namespace/}WssUsernameService
    enableTestPage: true
    enableWSDL: true

            JRFWssUsernamePort  http://host:port/j2wbasicPolicy/WssUsername
            enable: true
            enableREST: false
            enableSOAP: true
            maxRequestSize: -1
            loggingLevel: NULL
            security: oracle/wss_username_token_service_policy , enabled=true

           Attached policy or policies are valid; endpoint is secure.
```

9. Import the application PDD to the new server, or to the new environment, using the `importJRFWSApplicationPDD` command.

   ```
   importJRFWSApplicationPDD(application,pddJarFileName)
   ```

   For example, to import the PDD JAR created in step 6 to the cloned Managed Server, use the following command:

   ```
   wls:/wls-domain/serverConfig> importJRFWSApplicationPDD
   ('/wls-domain/ClonedManagedServer/j2wbasicPolicy','/tmp/j2wbasicPolicy-PDD-2010
   0115-145338.jar')
   application  /wls-domain/ClonedManagedServer/j2wbasicPolicy  PDD has been
   reset, please restart application now to uptake changes!
   ```

10. Restart the application and, optionally, verify the changes by executing the `listWebServices(None,None,true)` command again. For example:

    ```
    wls:/wls-domain/serverConfig> listWebServices(None,None,true)
    .
    .
    .
    /wls-domain/ClonedManagedServer/j2wbasicPolicy :
        moduleName=j2wbasicPolicy,
    moduleType=web,serviceName={http://namespace/}WssUsernameService
        enableTestPage: true
        enableWSDL: true

            JRFWssUsernamePort  http://host:port/j2wbasicPolicy/WssUsername
            enable: true
            enableREST: false
            enableSOAP: true
            maxRequestSize: -1
            loggingLevel: NULL
            security:oracle/wss_username_token_service_policy , enabled=true
            mtom : oracle/wsmtom_policy , enabled=true

           Attached policy or policies are valid; endpoint is secure.
    ```

11. Use the `savePddToAllAppInstancesInDomain` command to apply the policy changes (that you applied to a single server in step 9 using the `importJRFWSApplicationPDD` command) to all the server instances in the domain. This command is useful for propagating run-time policy changes to all servers in a cluster.

    ```
    savePddToAllAppInstancesInDomain(applicationName,pddJarFileName,restartApp=true
    )
    ```

    For example, to apply the same policy attachment to all server instances running the application in the domain, use the following command:

```
wls:/wls-domain/serverConfig>
savePddToAllAppInstancesInDomain('j2wbasicPolicy','/tmp/j2wbasicPolicy-PDD-2010
0115-145338.jar',true)
```

The `oracle/wsmtom_policy` is now attached to all server instances in the domain and the application is automatically restarted.

12. Optionally, execute the `listWebServices(None,None,true)` command to verify the changes were applied to all servers.

For more information about these deployment migration WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

16

Diagnosing Problems

This chapter contains the following sections:

16.1 Diagnosing Problems with Oracle WSM Policy Manager

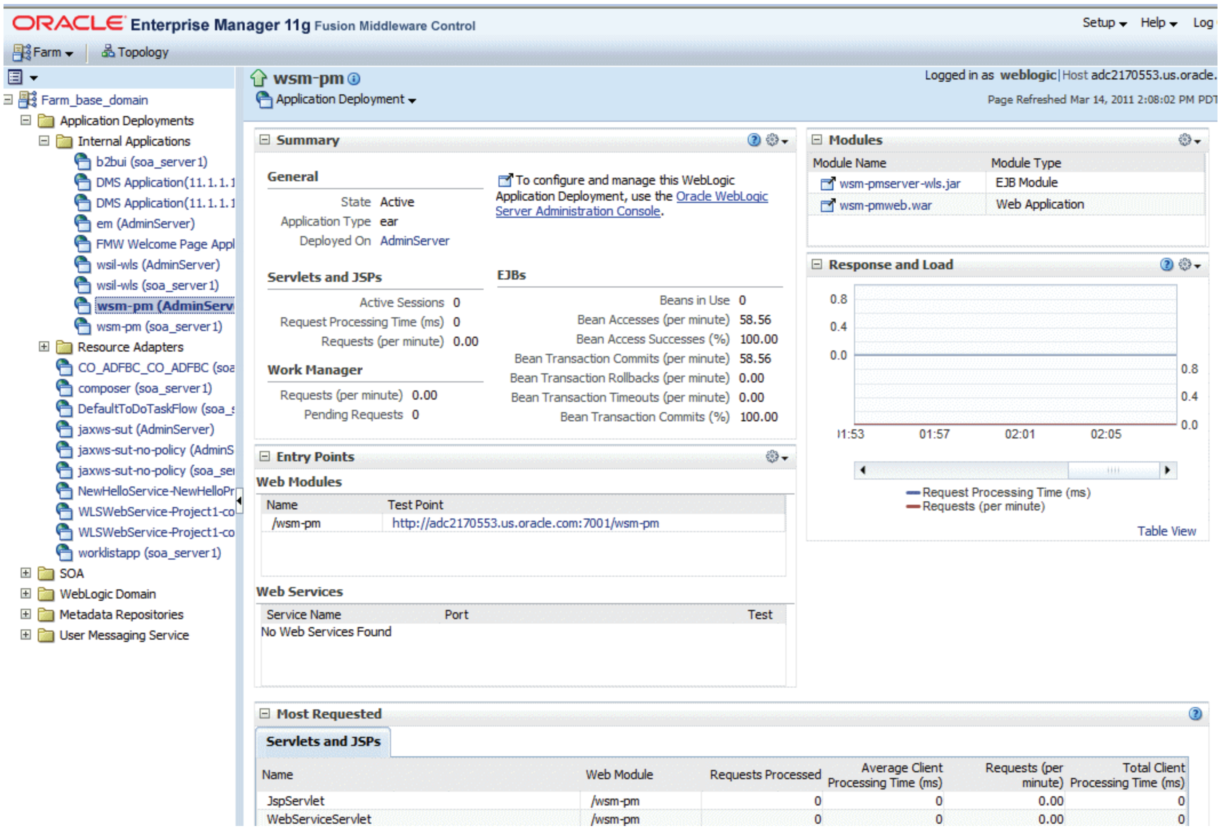
The Oracle WSM Policy Manager manages all Oracle WSM policies and needs to be running to use the Oracle WSM policy framework. You can check the current state of the Policy Manager and review its response time, load, and other data from the Oracle WSM Policy Manager page in Oracle Enterprise Manager Fusion Middleware Control.

To view the Oracle WSM Policy Manager page:

1. In the Navigator pane, expand **Application Deployments**.
2. Under Application Deployments, expand **Internal Applications**.
3. Select **wsm-pm**.

   The Oracle WSM Policy Manager home page is displayed.

Figure 16–1 Oracle WSM Policy Manager Page



Logged in as weblogic | Host adc2170553.us.oracle.

Page Refreshed Mar 14, 2011 2:08:02 PM PDT

General

State	Active
Application Type	ear
Deployed On	AdminServer

To configure and manage this WebLogic Application Deployment, use the Oracle WebLogic Server Administration Console.

Servlets and JSPs

Active Sessions	0
Request Processing Time (ms)	0
Requests (per minute)	0.00

Work Manager

Requests (per minute)	0.00
Pending Requests	0

EJBs

Beans in Use	0
Bean Accesses (per minute)	58.56
Bean Access Successes (%)	100.00
Bean Transaction Commits (per minute)	58.56
Bean Transaction Rollbacks (per minute)	0.00
Bean Transaction Timeouts (per minute)	0.00
Bean Transaction Commits (%)	100.00

Modules

Module Name	Module Type
wsm-pmserver-wls.jar	EJB Module
wsm-pmweb.war	Web Application

Web Modules

Name	Test Point
/wsm-pm	http://adc2170553.us.oracle.com:7001/wsm-pm

Web Services

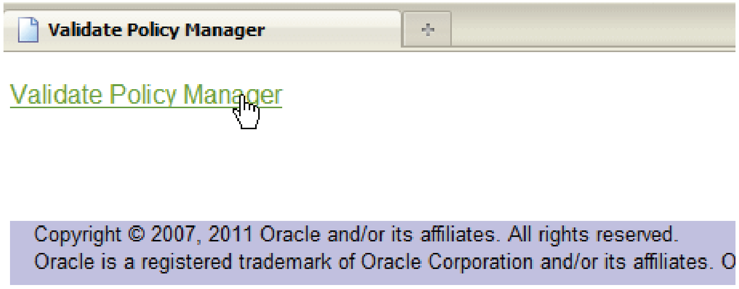
Service Name	Port	Test
No Web Services Found		

Most Requested — Servlets and JSPs

Name	Web Module	Requests Processed	Average Client Processing Time (ms)	Requests (per minute)	Total Client Processing Time (ms)
JspServlet	/wsm-pm	0	0	0.00	0
WebServiceServlet	/wsm-pm	0	0	0.00	0

4. From the Policy Manager page, you can perform one or more of the following tasks:
   - In the General area of the page, you can check the current state of the Policy Manager and identify the server to which it is deployed.
   - In the Response and Load section of the page, you can view the response time and current load. To view this information in tabular form, click **Table View**.
   - In the Entry Points section of the page, you can validate the connection to the Policy Manager. To do so, in the Web Modules table, click the Test Point URL for wsm-pm. On the Validate Policy Manager page, click the **Validate Policy Manager** link, as shown in Figure 16–2.

Figure 16–2 Validate Policy Manager Page



Copyright © 2007, 2011 Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. O

You can also access the Validator page in a Web browser using the following URL:

```
http://host:port/wsm-pm/validator
```

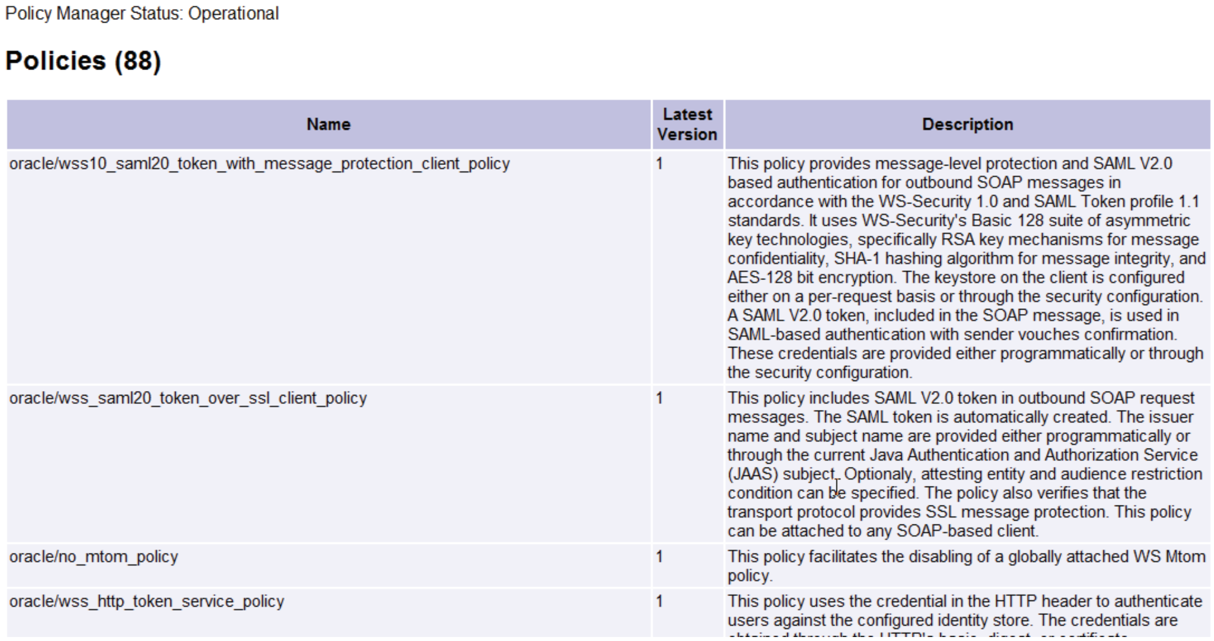
In this URL, *host* and *port* represent the host and port number on which the Policy Manager is running.

If the connection to the Policy Manager fails, an error message is displayed. If the connection to the Policy Manager is successful, the Policy Manager Validator page displays the following information:

- The status of the Policy Manager.
- The total number of Oracle WSM policies in the Oracle WSM Repository
- The name, latest version, and description of all the Oracle WSM policies in the Oracle WSM Repository.
- The total number of Oracle WSM assertion templates in the repository
- The name, latest version, and description of all the Oracle WSM assertion templates in the Oracle WSM Repository.
- The creation date and build label of the repository.

A sample Policy Manager Validator page is shown in Figure 16–3.

Figure 16–3 Policy Manager Validator Page



Policy Manager Status: Operational

Policies (88)

Name	Latest Version	Description
oracle/wss10_saml20_token_with_message_protection_client_policy	1	This policy provides message-level protection and SAML V2.0 based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 and SAML Token profile 1.1 standards. It uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. The keystore on the client is configured either on a per-request basis or through the security configuration. A SAML V2.0 token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation. These credentials are provided either programmatically or through the security configuration.
oracle/wss_saml20_token_over_ssl_client_policy	1	This policy includes SAML V2.0 token in outbound SOAP request messages. The SAML token is automatically created. The issuer name and subject name are provided either programmatically or through the current Java Authentication and Authorization Service (JAAS) subject. Optionaly, attesting entity and audience restriction condition can be specified. The policy also verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based client.
oracle/no_mtom_policy	1	This policy facilitates the disabling of a globally attached WS Mtom policy.
oracle/wss_http_token_service_policy	1	This policy uses the credential in the HTTP header to authenticate users against the configured identity store. The credentials are obtained through the HTTP's basic, digest, or certificate

For details about the Oracle WSM Repository, see Chapter 17, "Maintaining the Oracle WSM Repository."

16.2 Diagnosing Common Problems with Oracle WSM

The following sections describe some common problems you may encounter while using Oracle WSM, as well as possible solutions:

- Unable to Connect to the Policy Manager

- Key or Credential Store Error After an Application Invokes Web Service
- Trust Certificate Error After Application Invokes Web Service
- SAML Assertion Error Appears During Identity Propagation
- Policy Access Error After an Application Invokes Web Service
- Unable to Access User in Credential Store
- Authorization Error After an Application Invokes Web Service
- Timestamp Error After an Application Invokes Web Service
- Multiple Authentication Security Policy Error After an Application Invokes a Web Service
- STS Configuration Details Could Not Be Retrieved from Advertised WSDL

16.2.1 Unable to Connect to the Policy Manager

The following errors appear when you attempt to connect the Policy Manager:

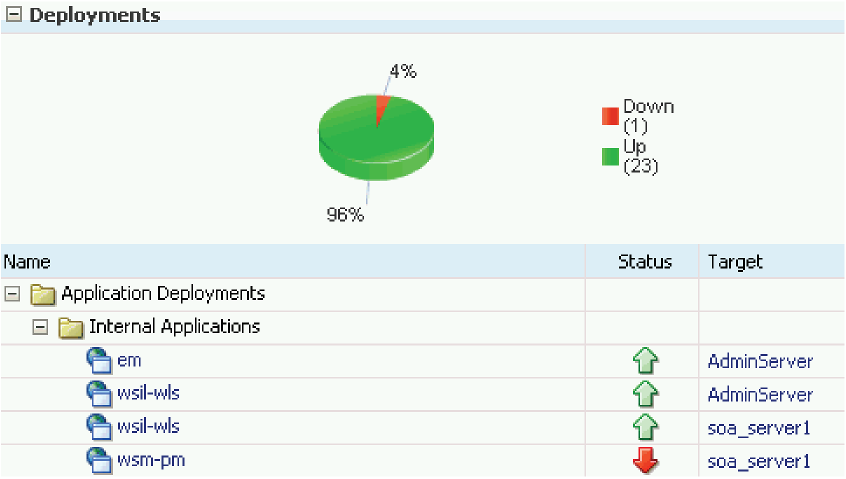
- `WSM-06157: The repository database is not configured correctly or not running.`
- `WSM-06160: The policy manager application has not been deployed or is not running.`
- `WSM-06161: The policy manager application has not been deployed.`
- `WSM-06162: The policy manager application is not running or is not configured correctly.`
- `WSM-06159: Cannot connect to the policy manager due to credential issue.`

Problem

The problem may be:

- The Policy Manager is down. You can determine if the Policy Manager is down as follows:
  - The state of the Policy Manager in the General area of the Oracle WSM Policy Manager home page, as described in "Diagnosing Problems with Oracle WSM Policy Manager" on page 16-1 is shown as shutdown.
  - The status of the wsm-pm internal application on the Farm home page in Enterprise Manager is Down, as shown in Figure 16–4. To access the Farm home page, select **Farm_em_domain** in the Navigator pane, or select **Home** from the Farm menu in the left-hand corner of the page.

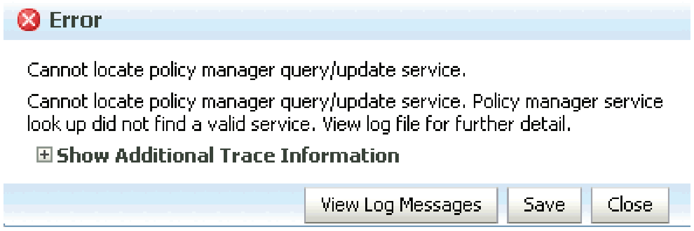
Figure 16–4 Oracle WSM Policy Manager Shutdown (Farm Page)



Name	Status	Target
Application Deployments		
Internal Applications		
em	↑	AdminServer
wsil-wls	↑	AdminServer
wsil-wls	↑	soa_server1
wsm-pm	↓	soa_server1

- An error dialog box similar to the following displays when you attempt to access the Oracle WSM policy management pages in Enterprise Manager. This error information is also written to the diagnostic log file, as described in "Reviewing Sample Logs" on page 16-26.

Figure 16–5 Error Message—Oracle WSM Policy Manager Unavailable



Cannot locate policy manager query/update service.

Cannot locate policy manager query/update service. Policy manager service look up did not find a valid service. View log file for further detail.

- The Policy Manager is targeted to an SSL server.

  Oracle Web Services Manager (WSM) supports an auto-discovery feature that it uses to locate and connect to an Oracle WSM Policy Manager within the same WebLogic domain. If the domain includes an SSL-configured server that has a Policy Manager deployed, the auto-discovery logic will connect to that Policy Manager and will not try to connect to any Policy Managers deployed on non-SSL servers. To ensure that the secure connection is maintained, the auto-discovery logic will not attempt to connect to a Policy Manager on a non-SSL server, even if the SSL-enabled server goes down. Therefore, even though there is a Policy Manager running, because it is running on a non-SSL enabled server, it is ignored and an error message is displayed.

- The credential required to access the Policy Manager is invalid or is not authorized.

- The repository may not be configured correctly.

Solution

If the Policy Manager is down:

Restart the wsm-pm application as described in "Starting and Stopping Applications" in *Oracle Fusion Middleware Administrator's Guide*.

If the Policy Manager is targeted to an SSL Server:

- Verify that the wsm-pm Policy Manager application is targeted to an SSL server. You can do so using the WebLogic Server Administration Console as described in "Target an Enterprise application to a server" in the *Oracle WebLogic Server Administration Console Help*.
- Verify that SSL has been configured correctly and that there are no SSL certificate issues. For additional information, see "Configuring Keystores for SSL" on page 10-36.
- If the SSL-enabled server is down, restart it, and the Policy Manager application, as described in "Starting and Stopping Oracle Fusion Middleware" in *Oracle Fusion Middleware Administrator's Guide*.
- If you want to use the Policy Manager on the non-SSL enabled server, untarget the Policy Manager application from the SSL-enabled server. For information about targeting applications to a server, see Managing Deployed Applications in *Deploying Applications to Oracle WebLogic Server*. To change the target server using the WebLogic Server Administration Console, see "Change target servers" in *Oracle WebLogic Server Administration Console Help*.

If there is a credential issue when attempting to access the Policy Manager:

By default, the Oracle WSM run time uses the OracleSystemUser account. If you are not using the default user accounts, you need to modify the configuration as described in "Modifying the Default User" on page 14-37.

If there is a problem with the repository configuration:

- Verify that the database and MDS schema are setup correctly. This configuration is performed as part of the installation process. For more information, see *Oracle Fusion Middleware Repository Creation Utility User's Guide*.
- Verify that the JDBC configuration is correct. The JDBC configuration is defined when you create the domain using the Oracle Fusion Middleware Configuration Wizard. For more information, see *Creating Domains Using the Configuration Wizard*.

16.2.2 Key or Credential Store Error After an Application Invokes Web Service

After an application invokes a Web service, a key store or credential store error such as the following appears:

- `WSM-00056: The key <alias_name> is not retrieved`
- `WSM-00256: The property "Keystore Sign Alias" is not set`

Problem

The problem may be:

- The alias for the signature key or encryption key in the Oracle WSM keystore configuration does not exist in the Oracle WSM keystore.
- The signature key, encryption key, or Oracle WSM keystore password is not synchronized between the keystore file and the keystore configuration for Oracle WSM. That is, at least one of the passwords does not have identical values in both locations.

Solution

To verify the alias for the signature key and encryption key in the Oracle WSM keystore configuration exist in the Oracle WSM keystore file:

1. Use Fusion Middleware Control to identify the alias for the signature key and encryption key in the Oracle WSM keystore configuration by performing the procedure in "Configuring the Oracle WSM Keystore" on page 10-11.
2. Verify the aliases identified in step 1 exist in the Oracle WSM keystore file. To do so, use the `keytool -list` command on the Oracle WSM keystore file, as described in step 4 of "Generating Private Keys and Creating the Java Keystore" on page 10-9. For detailed information about using the keytool utility, see the *keytool - Key and Certificate Management Tool* document at the following URL:

   `http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html`

   > **Note:** If you are unable to locate the document at the above URL, you can access it by searching for it on the Search Java SE Technical Documentation Web page at:
   >
   > `http://download.oracle.com/javase/search.html`

   - Ensure each alias is synchronized in both the keystore file and the Oracle WSM keystore configuration in the credential store. If they are not, you can edit the alias in the Oracle WSM keystore configuration by performing the procedure described in "Configuring the Oracle WSM Keystore" on page 10-11. You can edit the alias in the Oracle WSM keystore file using the `keytool -changealias` command.

     > **Note:** Before you edit an alias, be sure that doing so will not affect any other Web service.

   - If the alias for the signature key or encryption key does not exist in the Oracle WSM keystore file, add it as described in "Generating Private Keys and Creating the Java Keystore" on page 10-9.

To ensure that the signature key, encryption key, and Oracle WSM keystore file passwords are each synchronized in the keystore file and the keystore configuration for Oracle WSM:

1. Use `keytool` to reset the passwords in the Oracle WSM keystore file. Because the passwords are not visible, resetting them is the only method to ensure that they have identical respective values in both locations.
   - Use the `keytool -storepasswd` command to reset the Oracle WSM keystore file password.
   - Use the `keytool -keypasswd` command to reset the signature key password and encryption key password.
2. Use Fusion Middleware Control to reset the passwords in the Oracle WSM keystore configuration to the same respective values you set in step 1, as described in "Configuring the Oracle WSM Keystore" on page 10-11.

16.2.3 Trust Certificate Error After Application Invokes Web Service

After an application invokes a Web service, a trust certificate error such as the following appears:

```
WSM-00138: The path to the certificate is invalid due to exception
```

Problem

The problem may be, if the Web service is advertising its certificate in the Web Services Description Language (WSDL), the client may not be configured correctly to trust that certificate or its issuer.

Solution

To verify the client is configured to trust the Web service's certificate advertised in the WSDL or its issuer:

1. Verify the client keystore has either the certificate of the Web service or the certificate of its issuer.

   Use the `keytool -list` command to identify the certificates in the client keystore. If either of the certificates is missing from the client keystore, use the `keytool -importcert` command to add them.

   Refer to the *keytool - Key and Certificate Management Tool* document on the Java SE Technical Documentation Web site for more information about using `keytool`. You can access this document at the following URL:

   `http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html`

2. If the certificate is not published in the service's WSDL, verify that the value for the `keystore.recipient.alias` override property of the client policy is identical to the alias of the certificate in the Oracle WSM keystore file.

   For more information, see "Attaching Client Policies Permitting Overrides" on page 8-31.

16.2.4 SAML Assertion Error Appears During Identity Propagation

After an application attempts to propagate a user's identity by calling a different application using Oracle SOA, `InvalidSecurityToken`, `FailedAuthentication`, and `SAML assertion issuer` related errors appear.

Problem

The problem may be:

- The SAML issuer name for the SAML token is not configured or is configured incorrectly.
- The subject.precedence configuration override is set incorrectly.

Solution

To troubleshoot the SAML issuer name configuration:

Verify that the SAML Issuer Name that the client is using is among the issuers configured in the Oracle WebLogic Server domain. To do so, perform the steps described in "Adding an Additional SAML Assertion Issuer Name" on page 10-67.

If the SAML Issuer Name that the client is using is not configured as an issuer in the Oracle WebLogic Server domain, Oracle recommends changing the issuer name on the client by updating its saml.issuer.name override to one of the issuers configured in the domain.

If you cannot change the issuer name on the client, you can add its issuer name to the Oracle WebLogic Server domain by performing the steps in the "Adding an Additional SAML Assertion Issuer Name" on page 10-67.

> **Note:** If you make any changes to the issuers configured in the Oracle WebLogic Server domain, you must restart the Managed Server where Oracle WSM is deployed.

To troubleshoot the subject.precedence configuration override:

1. Set the subject.precedence override value in your current client policy to false to change the identity to a different user. By default, the subject.precedence override is set to true.
2. Set the appropriate Credential Store Framework key override on the client policy that contains the user name and password of the user you want to send to the service. If an entry for this user does not exist in the Credential Store Framework, you must add it. For more information, see "Configuring the Credential Store" on page 10-18
3. Ensure the appropriate Web Services Identity Permission is set for the client application by performing the steps in "Configuring Web Service Clients for Identity Switching" on page 10-78.

16.2.5 Policy Access Error After an Application Invokes Web Service

After an application attempts to invoke a Web service, a policy access error such as the following appears:

- `WSM-06156: The policy URI is missing, empty or contains invalid characters.`
- `WSM-06158: The referenced policy does not exist in the repository.`
- `WSM-02017: The document was not found in the repository.`

Problem

The problem may be:

- The policy URI is missing or the policy name is misspelled.
- The Policy Manager is down
- The policy does not exist in the repository
- The policy attachment is not in effect due to a cache delay.

Solution

To diagnose and solve policy access issues:

1. Verify that the Policy Manager is running as described in "Diagnosing Problems with Oracle WSM Policy Manager" on page 16-1 and "Unable to Connect to the Policy Manager" on page 16-4.
2. Verify that the mds-owsm datasource connection is reachable and available. For more information, see "Understanding and Managing Data Sources" in *Oracle Fusion Middleware Administrator's Guide*.
3. Verify that the policy exists in the Oracle WSM Repository by viewing the contents of the repository using the Policy Manager Validator page. For details about accessing the Validator page and viewing the contents of the repository, see "Diagnosing Problems with Oracle WSM Policy Manager" on page 16-1.

4. If the policy exists in the repository, verify that the policy URI is consistent with the policy URI in the repository.
5. If the policy does not exist in the Oracle WSM Repository, do one of the following:
   - For predefined policies:
     - Verify that the repository has been upgraded with all of the latest predefined policies using the `upgradeWSMPolicyRepository()` command. For more information, see `upgradeWSMPolicyRepository` in *WebLogic Scripting Tool Command Reference*.
     - Reset the contents of the repository using the `resetWSMPolicyRepository` command as described in "Rebuilding the Oracle WSM Repository" on page 17-7.
   - For a custom policy:
     - Import it into the repository as described in "Importing Web Service Policies" on page 7-7. For information on creating a custom policy, see "Creating Web Service Policies" on page 7-4.
6. Check if the user is in a role that has the right permission granted. To modify any roles or permissions, refer to "Modify the User's Group or Role" on page 14-38.
7. Verify the policy accessor and cache delay.

   The amount of time it takes for a policy attachment to take effect is determined by the Oracle WSM policy accessor and policy cache property settings. By default, this delay can be up to a maximum of 11 minutes. To reduce the amount of the delay, if necessary, you can tune the following cache property settings:

   - Policy Accessor

     `cache.refresh.initial`, default 600000 milliseconds (10 minutes)

     `cache.refresh.repeat`, default 600000 milliseconds (10 minutes)
   - Policy Cache

     `cache.tolerance`, default is 60000 milliseconds (1 minute)

   For details about tuning these properties, see "Configuring Platform Policy Properties" on page 14-15.

16.2.6 Unable to Access User in Credential Store

When Oracle WSM attempts to access a user in the credential store, an error such as the following occurs:

```
WSM-00015: The user name is missing
```

Problem

Oracle WSM cannot locate the user name in the credential store. This can be caused by any of the following:

- The credential map `oracle.wsm.security` does not exist in the credential store.
- The user is not listed in the map used by Oracle WSM.
- The csf key for the entry does not exist in the credential store.

Solution

Verify that the credential map `oracle.wsm.security` exists in the credential store. Oracle WSM only reads from this credential store map.

To determine if the `oracle.wsm.security` credential map exists in the credential store, refer to the procedure in "Configuring the Credential Store" on page 10-18.

If your application uses a credential map other than `oracle.wsm.security`, ensure that any users that Oracle WSM needs to access are duplicated in the `oracle.wsm.security` credential map.

16.2.7 Authorization Error After an Application Invokes Web Service

After an application attempts to invoke a Web service, an error such as the following appears:

```
java.security.AccessControlException: access denied
(oracle.wsm.security.WSFunctionPermission
```

Problem

Generally this is not really a problem rather intended behavior; that is, the system was unable to authorize the user for the action that the user is attempting. To debug check the calling server diagnostic log for the authorization error. The error may look similar to the following:

```
2011-01-06T22:15:43.691-08:00] [SalesServer_2] [ERROR] []
[oracle.jbo.server.svc.ServicePermissionCheckInterceptor_w2f8f5_Impl]
[tid: [ACTIVE].ExecuteThread: '7' for queue: 'weblogic.kernel.Default
(self-tuning)']
[userId: FMW_APPS_CRM_SELFSERVICE_ADF_APPID]
[ecid: 004aIPwzJDGE8TQRyaI7T00001WJ00EJ8f,0:1:3:1:11:0x5f5e189:6:1]
[WEBSERVICE_PORT.name: PartnerServiceSoapHttpPort] [APP: SalesApp#V2.0]
[J2EE_MODULE.name: partnerCenterCorePublicModel]
[WEBSERVICE.name: PartnerService] [J2EE_APP.name: SalesApp_V2.0]
[URI: /partnerCenterCorePublicModel/PartnerService] [[
java.security.AccessControlException: access denied
(oracle.wsm.security.WSFunctionPermission
http://xmlns.oracle.com/apps/partnerMgmt/partnerCenter/PartnerService#updatePartne
r invoke)
                at
java.security.AccessControlContext.checkPermission(AccessControlContext.java:323)
                at
java.security.AccessController.checkPermission(AccessController.java:546)
                at
oracle.jbo.server.svc.ServicePermissionCheckInterceptor.checkPermission(ServicePer
missionCheckInterceptor.java:103)
                at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

Solution

Pay careful attention to the following information in the log, which is shown in bold text in the example above.

1. The application stripe name – which is `SalesApp#V2.0` in the above log. Make sure it matches what you configured for your application. For information about how to configure the stripe name, see "Configuring the Servlet Filter and the EJB Interceptor" in *Oracle Fusion Middleware Application Security Guide*.
2. The permission grant, which is comprised of the following:
   a. Resource name, which is `http://xmlns.oracle.com/apps/partnerMgmt/partnerCenter/PartnerService#updatePartner` in the above log.
   b. Action, which is `invoke` in the above log.

Both of these pieces of information must be specified correctly in your permission grant. For more information, see "How Authorization Permissions Are Determined" on page 11-100.

If your application uses an LDAP-based authenticator and stores all roles in the LDAP, ensure that Oracle WSM can access the users and roles as described in "Modifying the Default User" on page 14-37.

16.2.8 Timestamp Error After an Application Invokes Web Service

After an application invokes a Web service, a timestamp or clockSkew error such as the following occurs:

```
WSM-00060: Error validating timestamp
```

Problem

The problem will either manifest itself as a timestamp validation or clockSkew error as shown below:

```
Caused By: FAULT CODE: InvalidSecurityToken FAULT MESSAGE: Found invalid condition
"on or after" in SAML assertion.
Current Time:Fri Feb 11 22:16:42 IST 2011, clockSkew:300000 milli seconds,
NotOnOrAfter Time:Fri Feb 11 14:21:42 IST 2011.
```

This problem usually happens if your server and client clocks are more than five minutes apart after they are converted to the same time zone.

Solution

Change your client or server clock in one of the following ways so that they are within five minutes, both set to the correct time:

- Adjust the clockSkew as described in "Tuning Web Service Security Policy Enforcement" on page 14-20.
- Set the system clock
- Use an ntp server to maintain the time

16.2.9 Multiple Authentication Security Policy Error After an Application Invokes a Web Service

After an application invokes a Web service, a multiple policy error (`WSM-01823`) appears in the log. This error appears, for example, if multiple authentication policies are attached to a subject.

Problem

More than one authentication policy was attached to a subject. This can happen if you have two policy sets that each attach an authentication policy to the same resource type, such as a Web service. For example, if you have two policy sets defined in the Oracle WSM Repository for your domain and one defines the policy scope as Domain("*domain_name*") and the other as Domain ("*"). The following listing illustrates an example of this scenario.

```
wls:/soa_domain/serverConfig> displayPolicySet('default-domain-ws-domain_gpa')

   Policy Set Details:
   -------------------
   Name:              default-domain-ws-domain_gpa
   Type of Resources:   Web Service Endpoint
   Scope of Resources:  Domain("soa_domain")
   Description:         Global policy attachments for Web Service Endpoint
resources.
   Enabled:             true
   Policy Reference:    security : oracle/wss11_saml_or_username_token_with_
message_protection_service_policy, enabled=true

wls:/soa_domain/serverConfig> displayPolicySet('default-domain-ws-domain')

   Policy Set Details:
   -------------------
   Name:                default-domain-ws-domain
   Type of Resources:   Web Service Endpoint
   Scope of Resources:  Domain("*")
   Description:         Global policy attachments for Web Service Endpoint
resources.
   Enabled:             true
   Policy Reference:    security : oracle/wss_saml_or_username_token_service_
policy, enabled=true
```

In this example, there are two policy sets with different names and different authentication policies pointing to the same resource type on the domain.

> **Note:** If the authentication policies attached to the subject are exact duplicates of each other, including any configuration overrides, the policy attachment is viewed as a duplicate and the configuration is valid.

Solution

To verify if you have multiple policy sets attempting to attach authentication policies:

1. Use the `listPolicySets()` command to display a list of the policy sets in the domain. For more information about this command, see "Displaying a List of Policy Sets Using WLST" on page 9-4.

2. Use the `listWebServices` or `listWebServiceClients` command, with the `detail` argument set to `true`, to view the endpoint (port) and policy details for all applications and composites in the domain, the secure status of the endpoints, any configuration overrides and constraints, and if the endpoints have a valid configuration.

   Using the policy sets defined in the example scenario, the output from the `listWebServices(detail=true)` command is displayed as follows. Note that the two policies in conflict and the policy sets with which they are associated are shown in bold text.

   ```
   wls:/soa_domain/serverConfig> listWebServices(detail=true)

   /soa_domain/jrfServer_admin/jaxws-sut-no-policy :
           moduleName=jaxws-service, moduleType=web,
   serviceName={http://namespace/}TestService
           enableTestPage: true
           enableWSDL: true

                   TestPort
   http://host.example.com:8344/jaxws-service/TestService
                   enable: true
                   enableREST: false
   ```

```
                enableSOAP: true
                maxRequestSize: -1
                loggingLevel: NULL
                Constraint: No Constraint
                        (global) security : oracle/wss11_saml_or_username_
token_with_message_protection_service_policy, enabled=true
                               /policysets/global/default-domain-ws-domain_gpa
: Domain("*")
                        (global) security : oracle/wss_saml_or_username_token_
service_policy, enabled=true
                                /policysets/global/default-domain-ws-domain :
Domain("*")
                Attached policy or policies are not valid.
                One or more attached policies are not compatible with endpoint
or other attached policy.
```

3. If a conflict is found, do one of the following:

   - Delete one of the policy sets as described in "Deleting Policy Sets" on page 9-32.
   - Disable one of the policy sets as described in "Enabling and Disabling a Policy Set" on page 9-31.

For more information, see "Determining the Secure Status of an Endpoint" on page 9-36.

16.2.10 STS Configuration Details Could Not Be Retrieved from Advertised WSDL

When a Web service configured with STS policies (`oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy` and `oracle/sts_trust_config_service_policy`) is invoked through a SOA composite reference, the following exception is thrown:

```
Unable to invoke endpoint URI "http://host:port/service/port" successfully
due to: oracle.fabric.common.PolicyEnforcementException: GenericFault :
generic error.
```

Problem

STS configuration details could not be retrieved from the advertised WSDL. The problem may occur when the STS policy configuration information is retrieved from a WSDL while the wsm-pm Policy Manager application is not running. The server might have cached this WSDL locally instead of fetching a fresh copy, and so the invocation will continue to fail since the STS configuration information is not present.

Solution

To retrieve the necessary STS configuration details:

1. Restart the Policy Manager application, as described in "Starting and Stopping Applications" in the *Oracle Fusion Middleware Administrator's Guide*. This ensures that the STS configuration details are advertised in the WSDL when the Web service is invoked through a SOA composite reference.
2. When using the Create Web Service dialog to design a SOA composite, verify that the copy wsdl and its dependent artifacts into the project check box is not selected. If this option is enabled, then deselect the check box and redeploy the SOA composite. For more information, see "Adding Service Binding Components" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

16.3 Diagnosing Policy Attachment Issues Using WLST

To ensure that there are no conflicts between directly-attached policies and policies attached globally using policy sets, use the following WLST commands:

- `listWebServices (detail="true")` — Displays a list of the Web services in a domain including endpoint configuration, the effective set of policies attached to each endpoint, the secure status of the endpoint, any configuration overrides and constraints, and if the endpoint has a valid configuration. For information about using this command, see "Viewing the Web Services in a Domain Using WLST" on page 6-2.
- `listWebServiceClients (detail="true")` — Displays a list of the Web service clients in a domain including endpoint configuration, the effective set of policies attached to each endpoint, the secure status of the endpoint, any configuration overrides and constraints, and if the endpoint has a valid configuration. For information about using this command, see "Viewing Web Service Clients", "Using WLST" on page 6-11

Note: An endpoint is considered secure if the policies attached to it (either directly, or externally using a policy set) enforce authentication, authorization, or message protection behaviors.

If your configuration includes policies attached globally using policy sets, you can view information about the policy sets using the following commands:

- `listPolicySets()` — Displays a list of the policy sets in the repository. For information about using this command, see "Displaying a List of Policy Sets Using WLST" on page 9-4.
- `displayPolicySet()` — Displays the configuration of a specific policy set. For information about using this command, see "Viewing the Configuration of a Policy Set", "Using WLST" on page 9-5.

To view the effective policies for an endpoint using Fusion Middleware Control, see "Viewing the Policies That are Attached to a Web Service" on page 8-1.

For more information about determining if the endpoint is secure and has a valid configuration, see "Determining the Secure Status of an Endpoint" on page 9-36.

Sample Valid Configuration Output with Globally and Directly Attached Policies

The following example shows sample output from the `listWebServices(detail=true)` command for a valid configuration. Because you can specify the priority of a global or directly attached policy (using the `reference.priority` configuration override), the `effective` field indicates if directly attached policies are in effect for the endpoint.

Note: To simplify endpoint management, all directly attached policies are shown in the output regardless of whether they are in effect for the endpoint. In contrast, only globally attached policies that are in effect for the endpoint are displayed.

```
/jrfServer_domain/jrfServer_admin/jaxws-sut :
        moduleName=jaxws-sut-service, moduleType=web,
serviceName={http://namespace/}TestService
        enableTestPage: true
        enableWSDL: true

                TestPort
http://host.example.com:9315/jaxws-sut-service/TestService
                enable: true
                enableREST: false
                enableSOAP: true
                maxRequestSize: -1
                loggingLevel: NULL
                management : oracle/log_policy, enabled=true
                security : oracle/wss_username_token_service_policy , enabled=true
, effective=false
                Constraint: No Constraint
                        (global) security : oracle/wss_saml_or_username_token_
service_policy, enabled=true

/policysets/global/all-domains-default-web-service-policies : Domain("*")
                                        reference.priority=1
                Constraint: HTTPHeader('VIRTUAL_HOST_TYPE','external')
                        (global) security : oracle/wss10_message_protection_
service_policy, enabled=true
                                /policysets/global/domainExternal : Domain("*")
                Attached policy or policies are valid; endpoint is secure.
```

Sample Valid Configuration Output with Directly Attached Policies Only

The following example shows sample output from the `listWebServices(detail=true)` command for a valid configuration. The directly attached policy is shown in bold text.

```
/jrfServer_domain/jrfServer_admin/jaxws-sut-no-policy :
        moduleName=jaxws-service, moduleType=web,
serviceName={http://namespace/}TestService
        enableTestPage: true
        enableWSDL: true

                TestPort
http://host.example.com:8344/jaxws-service/TestService
                enable: true
                enableREST: false
                enableSOAP: true
                maxRequestSize: -1
                loggingLevel: NULL
                security : oracle/wss_saml_or_username_token_service_policy,
enabled=true
                Attached policy or policies are valid; endpoint is secure.
```

16.4 Diagnosing Problems With a Domain Configuration using WLST

To ensure that there are no problems with the configuration of your domain, use the `checkWSMStatus` WLST command. The `checkWSMStatus` command returns the status of the policy manager (`wsm-pm`), the agent (`agent`), and the credential store and keystore configuration (`credstore`). The status of the components can be checked together or individually.

This command can be run after the provisioning of your WSM-protected Web service; there is no need to wait until after the first invocation.

Note: The Policy Manager (wsm-pm) application must be deployed and running for the checkWSMStatus command function correctly.

For more information on this command, see checkWSMStatus in *WebLogic Scripting Tool Command Reference*.

In the following example, the checkWSMStatus command returns a failure for the credential store because it is missing the key keystore-csf-key.

```
wls:/base_domain/serverConfig> checkWSMStatus('credstore')

Credential Store Configuration:



FAILED.
        Message(s):
             keystore.pass.csf.key : Property is configured and its value is
"keystore-csf-key".
                 Description: The "keystore.pass.csf.key" property points to the
CSF alias that is mapped to the username and password of the keystore.
Only the password is used; username is redundant in the case of the keystore.
             keystore-csf-key : Credentials not configured.

Credential Store Diagnostic Messages:
        Message(s):
                 The csf-key keystore-csf-key is not present in the credential
store.

 Perform the following steps to update the credential store (using WLST
commands):-
 1. connect()
 2. createCred(map="oracle.wsm.security", key="keystore-csf-key",
user="keystore-csf-key", password="<keystore-password>", desc="Keystore Password
CSF Key")
 NOTE:- All the above commands are based on the Domain level configurations.
The actual csf key may be overridden at runtime due to config override.
See Documentation for more details.
false
```

In the following example, the checkWSMStatus command returns the status of the credential store and keys, the policy manager, and the enforcement agent for the domain base_domain.

```
wls:/base_domain/serverConfig> checkWSMStatus()

Credential Store Configuration:

PASSED.
        Message(s):
             keystore.pass.csf.key : Property is configured and its value is
"keystore-csf-key".
                 Description: The "keystore.pass.csf.key" property points to the
 CSF alias that is mapped to the username and password of the keystore.
Only the password is used; username is redundant in the case of the keystore.
             keystore-csf-key : Credentials configured.
             keystore.sig.csf.key : Property is configured and its value is
"sign-csf-key".
                 Description: The "keystore.sig.csf.key" property points to the
 CSF alias that is mapped to the username and password of the private key that is
```

```
used for signing.
             sign-csf-key : Credentials configured.
             Sign Key : Key configured.
                 Alias - orakey
             Sign Certificate : Certificate configured.
                 Alias - CN=weblogic, OU=Orakey Test Encryption Purposes Only,
O=Oracle, C=US
                 Expiry - June 28, 2020 11:17:12 AM PDT
             keystore.enc.csf.key : Property is configured and its value is
"enc-csf-key".
                 Description: The "keystore.enc.csf.key" property points to the
 CSF alias that is mapped to the username and password of the private key that is
used for decryption.
             enc-csf-key : Credentials configured.
             Encrypt Key : Key configured.
                 Alias - orakey
             Encrypt Certificate : Certificate configured.
                 Alias - CN=weblogic, OU=Orakey Test Encryption Purposes Only,
O=Oracle, C=US
                 Expiry - June 28, 2020 11:17:12 AM PDT

Policy Manager:

PASSED.
        Message(s):
             OWSM Policy Manager connection state is OK.
             OWSM Policy Manager connection URL is "t3://host:7001".

Enforcement Agent:

PASSED.
        Message(s):
             Enforcement is successful.
             Service URL: http://host:7001/Diagnostic/DiagnosticService?wsdl
```

16.5 Diagnosing Common Oracle WSM Exceptions for WS-Trust Use Cases

Table 16–1lists the common Oracle WSM exceptions and errors that can occur during an end-to-end WS-Trust use case scenario. The probable cause and recommended solutions are also provided. For details about how to configure the supported WS-Trust use cases, see the following topics:

- "WS-Trust Policies and Configuration Steps" on page 10-98
- "Examples Using WS-Trust with OpenSSO STS" on page 10-111

Table 16–1 Common Oracle WSM Exceptions and Errors for WS-Trust Use Cases

Exception/Error	Possible Cause	Solution
`WSM-00015: The user name is missing.`	1. The `sts.auth.user.csf.key` configuration property may not be overridden in the STS issued token client policy. 2. If the property has been overridden, the CSF key may not be available in the credential store or the override may not be specifying the correct value.	Override the `sts.auth.user.csf.key` property in the STS issued token client policy with the correct value from the credential store.
`javax.net.ssl.SSLHandshakeException: PKIX path validation failed: java.security.cert.CertPathValidatorException: signature check failed`	When communicating with any service over an SSL channel, a valid SSL certificate for the service must be available in the trusted keystore for the JRE distribution being used in the environment. In most cases, the SSL certificate is found in the following directory: `JAVA_HOME/jre/lib/security/cacerts` This exception can occur due to either of the following: - The SSL certificate for the service could not be found in the trusted keystore. - The SSL certificate present in the trusted keystore may have expired.	Ensure that a valid SSL certificate for the service has been imported into the trusted keystore at `JAVA_HOME/jre/lib/security/cacerts`.For more information, see "Configuring Keystores for SSL" on page 10-36.
`WSM-00323 : STS ISSUER_ADDRESS obtained from WSDL is null. Local STS configuration is also not available.`	This exception occurs because both the client and the service do not have an STS trust config policy attached. For a simple WS-Trust use case, the STS trust config policy must be attached to either the client or the service application. Because there are no policies attached, the client does not know which STS to communicate with to get the SAML token.	Attach an STS trust config policy to the client or service application as required for your configuration. For more information, see "WS-Trust Policies and Configuration Steps" on page 10-98.
`FailedAuthentication: Security Token cannot be authenticated: Error in receiving the request: oracle.wsm.security.SecurityException: WSM-00062 : The path to the certificate used for the signature is invalid.`	The web service provider or Relying Party is not able to validate the Issuer's signature on the incoming SAML token.	Make sure that you have imported the issuer's public key/certificate into the JKS/KSS keystore that the service is using. For detailed procedures, see "Configuring Keystores for Message Protection" on page 10-9.

`InvalidSecurityToken : The security token is not valid. SAML assertion issuer name is invalid.`
`WSM-00376 : SAML token authentication failed for issuer "<issuer name>".`
	The SAML assertion issuer name is not configured in the trusted issuers list in the domain in which the Relying Party service is deployed.	Add the issuer to the list of trusted issuers in the domain in which the service is running.
		For detailed procedures, see "Defining Trusted Issuers and a Trusted DN List for Signing Certificates" on page 14-23.

`WSM-00231: Cannot find client compatible policy for STS  <STS WSDL URI>, port name <STS port name>"`	This exception can be thrown when a third party STS server is protected using a policy that does not have a compatible client policy in Oracle WSM. Any STS endpoint that the client is trying to communicate with is protected with a security policy. Oracle STS uses Oracle WSM which provides compatible client and service policies. In this case, the client should not have any trouble finding the corresponding client policy.	The Oracle WSM trust client has been tested with most common STS servers so it is unlikely that this exception will occur. In the event that this exception is thrown, a possible workaround is to attach a new or cloned version of the `oracle/sts_trust_config_client_policy` on the client side and configure it with the client policy to be used to communicate with the STS. For details, see "Manually Configuring the STS Config Policy From the Web Service Client: Main Steps" on page 10-107.

16.6 Diagnosing Problems with the Oracle WSM RESTful Client Filter

Once you have completed steps described in "Securing RESTful Web Service Clients" in *Using the Jersey JAX-RS Reference Implementation* to register the Oracle WSM RESTful client filter, the Oracle WSM polices that protect your RESTful client should be enforced. If they are not being enforced, ensure that there are no other JARs that define the `weblogic.jaxrs.api.client.Client` class and that are overriding the `wls-rest-client.jar` shared library in the CLASSPATH.

If you developed your RESTful client using JDeveloper, ensure that you disable the deployment of the Jersey JAX-RS Reference Implementation (RI) with your application by disabling the Deployed by Default property. For more information, see "Adding the Jersey JAX-RS Reference Implementation to Your Project" in "Developing RESTful Web Services" in the Oracle JDeveloper Online Help.

16.7 Diagnosing Problems Using Logs

Oracle Fusion Middleware components, including Web services, generate log files containing messages that record all types of events, including startup and shutdown information, errors, warning messages, access information on HTTP requests, and so on. Each log message includes specific information such as time, component ID, and user to assist you in pinpointing and diagnosing problems that arise.

You can review log messages to diagnose problems with specific components, such as Web services. There are two categories of log files that you can reference to assist in diagnosing problems with Web services:

- **Diagnostic logs—**Enable you to access diagnostic data about specific feature components in Oracle Fusion Middleware. For more information, see "Using Diagnostic Logs for Web Services" on page 16-20.

  There is a set of predefined diagnostic loggers. You can configure your own diagnostic logger, as described in "Configuring Log Files for a Web Service" on page 16-28.

- **Message logs—**Enable you to view elements of the SOAP message request. You control message log creation using policies. For more information, see "Using Message Logs for Web Services" on page 16-25.

For more information about logging in Oracle Fusion Middleware, see "Managing Log Files and Diagnostic Data" in *Oracle Fusion Middleware Administrator's Guide*.

The following sections describe how to use diagnostic and message logs to diagnose problems. A set of sample logs is provided at the end of this section.

- Using Diagnostic Logs for Web Services
- Using Message Logs for Web Services
- Reviewing Sample Logs

16.7.1 Using Diagnostic Logs for Web Services

Diagnostic logs enable you to access diagnostic data about specific feature components in Oracle Fusion Middleware.

The following sections describe how to view and manage diagnostic log files:

- Setting the Log Level for Diagnostic Logs
- Viewing Diagnostic Logs

- Filtering Diagnostic Logs
- Logging Oracle WSM Debug Messages

16.7.1.1 Setting the Log Level for Diagnostic Logs

You set the logging level for Web service and Oracle WSM components at the WebLogic Server level, using the Log Configuration page.

In addition, you can override the log levels set at the server level for a specific Web service endpoint from the Web Service Endpoint page. The logging level set at the Web service endpoint level must be "finer grained" than the level set at the WebLogic Server level. Otherwise, the logging level set at the WebLogic Server level will be used.

The following procedures describe how to set the log level for diagnostic logs at the WebLogic Server and Web service endpoint levels. For more information, see "Setting the Level of Information Written to Log Files" in *Oracle Fusion Middleware Administrator's Guide*.

To set the log level for diagnostics logs at the WebLogic Server level:

1. Navigate to the WebLogic Server for which you want to configure a logger.

   a. In the navigator pane, expand **WebLogic Domain**.

   b. Expand the domain.

   c. Select the desired server from the list.

      The WebLogic Server home page is displayed.

2. From the **WebLogic Sever** menu, select **Logs > Log Configuration**.

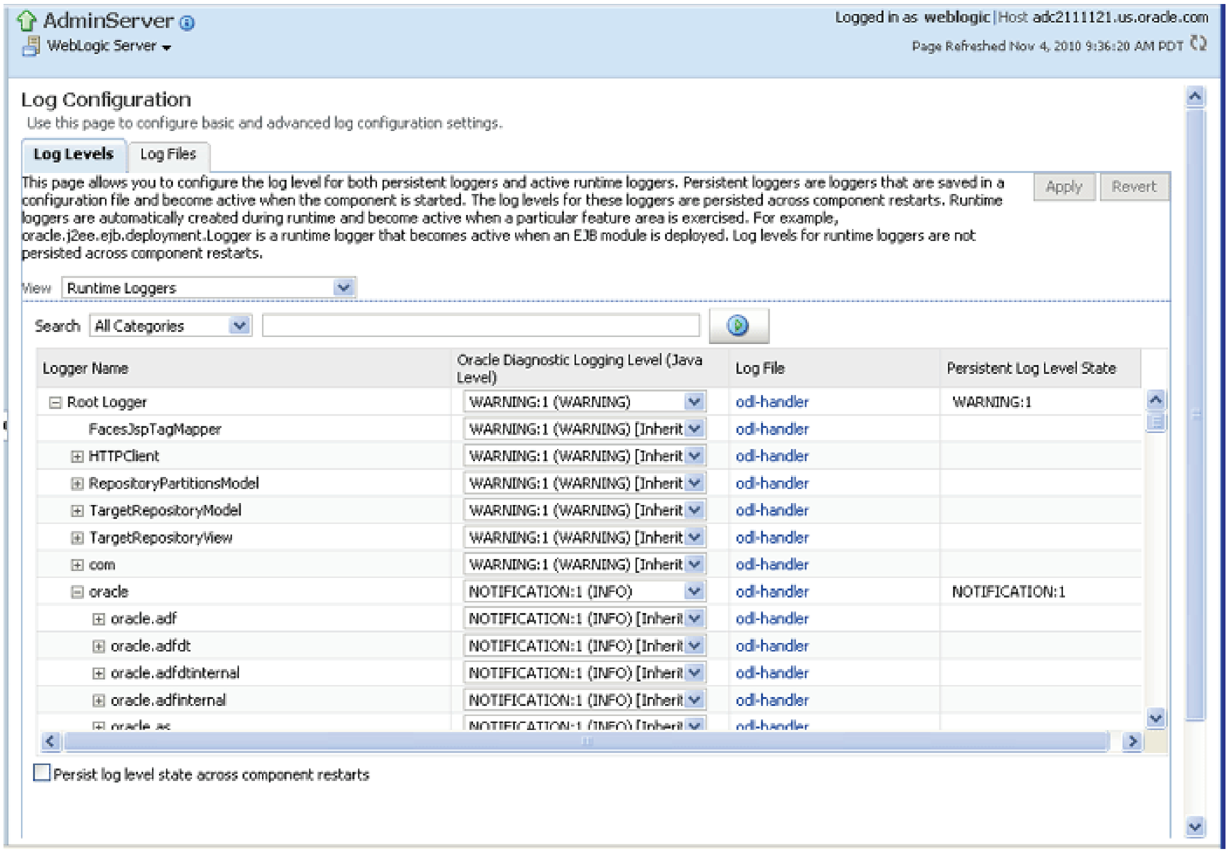
   The Log Configuration page is displayed.

3. Select the **Log Levels** tab.

   The list of loggers is displayed, as shown in Figure 16–6.

   The Log Levels page shows the name of the logger, the current logging level, which you can edit, and the associated log file (for example, olh-handler). For information about configuring the log files, see "Configuring Log Files for a Web Service" on page 16-28.

Figure 16–6 Log Levels Page



AdminServer
WebLogic Server

Logged in as weblogic | Host adc2111121.us.oracle.com
Page Refreshed Nov 4, 2010 9:36:20 AM PDT

Log Configuration
Use this page to configure basic and advanced log configuration settings.

This page allows you to configure the log level for both persistent loggers and active runtime loggers. Persistent loggers are loggers that are saved in a configuration file and become active when the component is started. The log levels for these loggers are persisted across component restarts. Runtime loggers are automatically created during runtime and become active when a particular feature area is exercised. For example, oracle.j2ee.ejb.deployment.Logger is a runtime logger that becomes active when an EJB module is deployed. Log levels for runtime loggers are not persisted across component restarts.

View Runtime Loggers

Search All Categories

Logger Name	Oracle Diagnostic Logging Level (Java Level)	Log File	Persistent Log Level State
⊟ Root Logger	WARNING:1 (WARNING)	odl-handler	WARNING:1
FacesJspTagMapper	WARNING:1 (WARNING) [Inherit	odl-handler	
⊞ HTTPClient	WARNING:1 (WARNING) [Inherit	odl-handler	
⊞ RepositoryPartitionsModel	WARNING:1 (WARNING) [Inherit	odl-handler	
⊞ TargetRepositoryModel	WARNING:1 (WARNING) [Inherit	odl-handler	
⊞ TargetRepositoryView	WARNING:1 (WARNING) [Inherit	odl-handler	
⊞ com	WARNING:1 (WARNING) [Inherit	odl-handler	
⊟ oracle	NOTIFICATION:1 (INFO)	odl-handler	NOTIFICATION:1
⊞ oracle.adf	NOTIFICATION:1 (INFO) [Inherit	odl-handler	
⊞ oracle.adfdt	NOTIFICATION:1 (INFO) [Inherit	odl-handler	
⊞ oracle.adfdtinternal	NOTIFICATION:1 (INFO) [Inherit	odl-handler	
⊞ oracle.adfinternal	NOTIFICATION:1 (INFO) [Inherit	odl-handler	
⊞ oracle.as	NOTIFICATION:1 (INFO) [Inherit	odl-handler	

☐ Persist log level state across component restarts

4. Expand **Root Logger**.
5. Expand **oracle**.
6. Set the logging level for one or more of the following components:
   - oracle.webservices—Web service components.
   - oracle.wsm—Oracle WSM components.

   You can fine tune the logging level by expanding either of the above components and specifying the logging level at the subcomponent level.

   To change the logging level for a logger, navigate to the logger in the Logger Name column and select the desired logging level from the **Oracle Diagnostic Logging Level (Java Level)** drop-down menu.

   For example, select TRACE:32 from the drop-down menu associated with the oracle.wsm logger.

   By default, the logging levels are inherited from the parent and set to NOTIFICATION: 1 (INFO) for the Web service and Oracle WSM components and subcomponents.
7. Click **Apply** to store the new logging level.

To set the log level for diagnostic logs at the Web service endpoint level:

1. Navigate to the Web Service Endpoint page, as described in "Viewing the Details for a Web Service Endpoint" on page 6-7.
2. Click the **Configuration** tab.

3. Set the **Logging Level** field to one of the following settings: Severe, Warning, Information, Configuration, Fine, Finer, Finest or NULL.

> **Note:** You can also set the log level at the Web service endpoint using the `setWebServiceConfiguration` WLST command. Set the `loggingLevel` property of the `itemProperties` argument to one of the following settings: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, or NULL. For details about using this command, see "setWebServiceConfiguration" in *WebLogic Scripting Tool Command Reference*.

16.7.1.2 Viewing Diagnostic Logs

You can view the diagnostic log files for an ADF and WebCenter Web service endpoint from the Log Messages page.

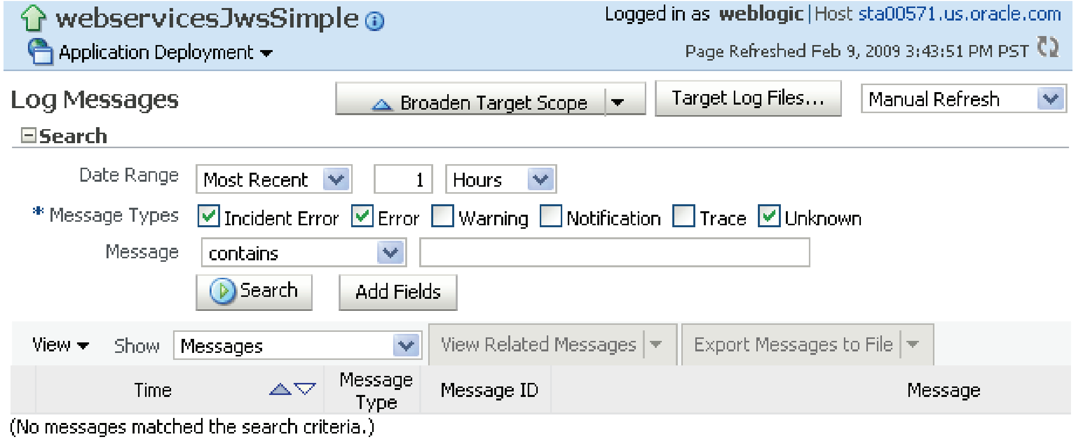
To view diagnostic logs for a Web service endpoint:

Navigate to the Web Service Endpoint page, as described in "Viewing the Details for a Web Service Endpoint" on page 6-7, and in the Quick Links section of the Web Services Endpoint page (top right), click **Diagnostic Logs.**

> **Note:** You can view a summary of all faults incurred by the Web services in your application. For more information, see "Monitoring the Performance of Web Services" on page 13-1.

The Log Messages page is displayed, as shown in the following figure.

Figure 16–7 Log Messages Page



webservicesJwsSimple

Application Deployment

Logged in as **weblogic** | Host sta00571.us.oracle.com

Page Refreshed Feb 9, 2009 3:43:51 PM PST

Log Messages

Search

Date Range: Most Recent, 1, Hours

* Message Types: ☑ Incident Error ☑ Error □ Warning □ Notification □ Trace ☑ Unknown

Message: contains

Time	Message Type	Message ID	Message

(No messages matched the search criteria.)

Click on a message in the message area to view more details at the bottom of the page. If desired, you can export a message to a text, XML, or CSV file by selecting the messages on the list and clicking **Export Messages to File**.

You can control the message content displayed using the following controls:

- **Search**—Modify the search criteria. For more information, see "Filtering Diagnostic Logs" on page 16-24.

- **View menu**—Select the columns to display in the table. Click on a particular column to sort contents up or down.
- **Show menu**—Group messages by type or ID, or view them in chronological order.
- **View Related Messages—**View messages related to those selected on the list.
- **Broaden Target Scope—**Broaden the scope of messages displayed. You can broaden the scope to include all messages for the domain, WebLogic Server, or Farm.
- **Refresh menu**—Specify an automatic or manual refresh rate.

To view the contents of a generated log file:

- Click the log file icon associated with a message to view the contents of that log file.
- Click **Target Log Files...** to display the Log Files page and view or download the contents of all generated log files.

For more information, see "Viewing and Searching Log Files" in *Oracle Fusion Middleware Administrator's Guide*.

16.7.1.3 Filtering Diagnostic Logs

By default, the Log Messages page displays a summary of diagnostic messages logged over the last hour.

To filter diagnostic logs:

1. Filter the messages that are displayed by updating the search criteria using the following fields:
   - **Date Range**—Set the date range to one of the following:
     - Most Recent—Set the amount of time to define the duration.
     - Time Interval—Set the start and end dates to define the interval.
   - **Message Types—**Select the message types that you want to display.
   - **Add Fields**—Add other message fields to your search criteria, such as Message ID, Component, and so on.
2. Click **Search** once you have set the fields, as desired.

   The messages area is updated with the filtered results.

For more information, see "Viewing and Searching Log Files" in *Oracle Fusion Middleware Administrator's Guide*.

16.7.1.4 Logging Oracle WSM Debug Messages

To debug Oracle WSM, pass one of the following properties when starting WebLogic Server, as required. For more information, see "Starting and Stopping Servers" in *Managing Server Startup and Shutdown for Oracle WebLogic Server*.

> **Note:** Enabling one or more of these properties may negatively impact performance for very large messages. When enabled, Oracle WSM creates temporary buffers which will result in additional load on the Java garbage collector.

Table 16–2 Startup Properties for Logging Oracle WSM Debug Messages

Startup Property	Description
`-Dxml.debug.verify=true`	Logs the sequence of bytes produced during a signature verification failure. Verification errors are output to `stderr` and the diagnostic log file when the log level is set to at least ERROR.
`-Dxml.debug.digest=true`	Verifies that the sequence of bytes produced during signature generation canonicalization and signature verification match. Verification errors are output to `stderr` and the diagnostic log file when the log level is set to at least FINE.
`-Dxml.debug.decrypt`	Logs the sequence of bytes produced following a decryption failure before XML parsing. Verification errors are output to `stderr` and the diagnostic log file.

16.7.2 Using Message Logs for Web Services

Message logs enable you to access the contents of the SOAP message requests and responses for ADF and WebCenter Web services and clients. Messages logs are stored in a log file separate from the diagnostic messages, by default.

The following sections describe how to view and manage message log files:

- Configuring Message Logs
- Viewing Message Logs
- Filtering Message Logs

16.7.2.1 Configuring Message Logs

You configure message logs for a Web service or client in one of the following ways:

- Attach a policy that contains a logging assertion to the Web service or client.

  There is one predefined logging assertion template: oracle/security_log_template, described in "oracle/security_log_template" on page C-195. This template is configured to log the entire SOAP message for the Web service request and response. By default, all predefined Web service security policies use this logging assertion to capture the entire SOAP message before and after the primary security assertion is executed. By default, the log assertion is not enforced. You must enable it in order for the SOAP message to be logged in message logs, as described in "Enabling or Disabling Assertions Within a Policy" on page 7-25. It is recommended that the logging assertion be enabled for debugging and auditing purposes only.

- Attach the oracle/log_policy policy to the Web service or client. For more information, see "oracle/log_policy" on page B-41.

- Create your own logging policy or assertion template to further refine the elements of the SOAP message that are logged for the Web service request and response.

  For example, you may wish to view only the SOAP body of the request message. To create a new policy, following the procedure described in "Creating Web Service Policies" on page 7-4. You may wish to create a copy of the oracle/security_log_template assertion template and configure it for use in the new policy. For more information about creating a new assertion template, see "Creating an Assertion Template" on page 7-9.

16.7.2.2 Viewing Message Logs

You can view the message log files for an ADF and WebCenter Web service endpoint from the Log Messages page.

To view message logs for a Web service endpoint:

Navigate to the Web Service Endpoint page, as described in "Viewing the Details for a Web Service Endpoint" on page 6-7, and in the Quick Links section of the Web Services Endpoint page (top right), click **Message Logs.**

The Log Messages page is displayed, similar to Figure 16–7. For more details about the contents of the Log Messages page, see "Viewing Diagnostic Logs" on page 16-23.

16.7.2.3 Filtering Message Logs

By default, the Log Messages page displays a summary of SOAP messages logged over the last hour. You can filter the messages that are displayed by updating the search criteria. The process is the same as for diagnostic logs; for more information, see "Filtering Diagnostic Logs" on page 16-24.

By default, the Component and Module message fields are included as part of the Search criteria for message logs. The Component field is set to the WebLogic Server name; the Module field is set to oracle.wsm.msg.logging, which is the name of the message logging component.

16.7.3 Reviewing Sample Logs

The following sections provide excerpts from sample logs, demonstrating how to diagnose specific problems using the log entries.

- Sample Log: Oracle WSM Policy Manager Not Available
- Sample Log: Security Keystore Not Configured
- Sample Log: Certificate Not Available

16.7.3.1 Sample Log: Oracle WSM Policy Manager Not Available

The following sample log excerpt indicates that the Oracle WSM Policy Manager is down. To resolve this issue, restart the wsm-pm application, as described in "Starting and Stopping Applications Using" in *Oracle Fusion Middleware Administrator's Guide*.

```
2009-02-16 16:21:28,029 [[ACTIVE] ExecuteThread: '4' for queue:
 'weblogic.kernel.Default (self-tuning)']
 ERROR policymgr.PolicyManagerModelBean logp.251 -
 Service lookup failed with URL:t3://host.example.com:7001/wsm-pm
 oracle.wsm.policymanager.PolicyManagerException: WSM-02118 :
 The query service cannot be created.
...
```

16.7.3.2 Sample Log: Security Keystore Not Configured

The following sample log excerpt indicates that an Oracle WSM security policy with message protection was applied, but the keystore was not configured. To resolve this security fault, configure the keystore, as described in "Configuring Keystores for Message Protection" on page 10-9.

```
Feb 16, 2009 5:29:56 PM oracle.wsm.common.logging.WsmMessageLogger logSevere
 SEVERE: The specified Keystore file /scratch/sbollapa/stage131/user_
projects/domains/sai131_domain/config/fmwconfig/default-keystore.jks
```

```
 cannot be found; it either does not exist or its path is not included in the
 application classpath.
 Feb 16, 2009 5:29:56 PM oracle.wsm.common.logging.WsmMessageLogger logSevere
 SEVERE: Keystore is not properly configured in jps config.
 Feb 16, 2009 5:29:56 PM oracle.wsm.common.logging.WsmLogUtil log
 SEVERE: failure in OWSM Agent processRequest, category=security,
 function=agent.function.client, application=default, composite=pe3test3,
 modelObj=Service1, + policy=null, policyVersion=null, assertionName=null
 oracle.wsm.common.sdk.WSMException: WSM-00101 : The specified Keystore file
 /scratch/sbollapa/stage131/user_projects/domains/sai131_
domain/config/fmwconfig/default-keystore.jks cannot be found;
 it either does not exist or its path is not included in the application
classpath.
...
```

16.7.3.3 Sample Log: Certificate Not Available

The following sample log excerpt indicates that an Oracle WSM security policy with message protection was applied that required a security certificate that was not available in the keystore. To resolve this security fault, configure the keystore with a certificate, as described in "Obtaining a Trusted Certificate and Importing it into the Keystore" on page 10-15.

```
[2009-04-15T04:07:02.821-07:00] [jrfServer] [ERROR] [WSM-000062]
 [oracle.wsm.resources.security] [tid: [ACTIVE].ExecuteThread: '0' for
queue: 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>]
 [ecid: 0000I2dTFG7DScT6uBe9UH19tRyv000000,0:1] [WEBSERVICE_PORT.name:
 NonCAAsCAMessageProtectionPolicyPort] [APP: jaxwsservices]
 [J2EE_MODULE.name: NonCAAsCAMessageProtectionPolicy] [WEBSERVICE.name:
 NonCAAsCAMessageProtectionPolicyService] [J2EE_APP.name: jaxwsservices]
 [arg: oracle.wsm.security.SecurityException: WSM-00062 :
 The path to the certificate used for the signature is invalid.]

[2009-04-15T04:07:02.810-07:00] [jrfServer] [NOTIFICATION] []
 [oracle.wsm.security.policy.scenario.processor.Wss11X509TokenProcessor]
 [tid: [ACTIVE].ExecuteThread: '0' for queue: 'weblogic.kernel.Default
 (self-tuning)'] [userId: <anonymous>]
[ecid: 0000I2dTFG7DScT6uBe9UH19tRyv000000,0:1]
[WEBSERVICE_PORT.name: NonCAAsCAMessageProtectionPolicyPort]
[APP: jaxwsservices] [J2EE_MODULE.name: NonCAAsCAMessageProtectionPolicy]
 [WEBSERVICE.name: NonCAAsCAMessageProtectionPolicyService] [J2EE_APP.name:
 jaxwsservices] Certificate path validation failed for signing certificate

[2009-04-15T04:07:02.821-07:00] [jrfServer] [ERROR] [WSM-00006]
 [oracle.wsm.resources.security] [tid: [ACTIVE].ExecuteThread: '0' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>]
[ecid: 0000I2dTFG7DScT6uBe9UH19tRyv000000,0:1] [WEBSERVICE_PORT.name:
 NonCAAsCAMessageProtectionPolicyPort] [APP: jaxwsservices]
[J2EE_MODULE.name: NonCAAsCAMessageProtectionPolicy] [WEBSERVICE.name:
 NonCAAsCAMessageProtectionPolicyService] [J2EE_APP.name: jaxwsservices]
[arg: oracle.wsm.security.SecurityException: WSM-00062 : The path to the
 certificate used for the signature is invalid.] Error in receiving the request:
 oracle.wsm.security.SecurityException: WSM-00062 : The path to the certificate
 used for the signature is invalid.
```

16.8 Configuring Log Files for a Web Service

To further organize your logging data, you can configure the log files for a Web service. You can configure log files for SOA, ADF, and Web Center services.

The following table defines the default log files that are relevant to Oracle WSM.

Table 16–3 Default Log Files for Oracle WSM

Default Log File	Description
odl-handler	Logs general diagnostic data for the Java EE components in the server.
owsm-message-handler	Logs SOAP messages as per Oracle WSM logging policies.

The following procedure describes how to set the log level for diagnostic logs at the WebLogic Server and Web service endpoint levels.

For more information about using Fusion Middleware Control or WLST to set the log levels, see "Setting the Level of Information Written to Log Files" in *Oracle Fusion Middleware Administrator's Guide*.

To configure the log files for a Web service:

1. Navigate to the WebLogic Server for which you want to configure a logger.

   a. In the navigator pane, expand **WebLogic Domain**.

   b. Expand the domain.

   c. Select the desired server from the list.

      The WebLogic Server home page is displayed.

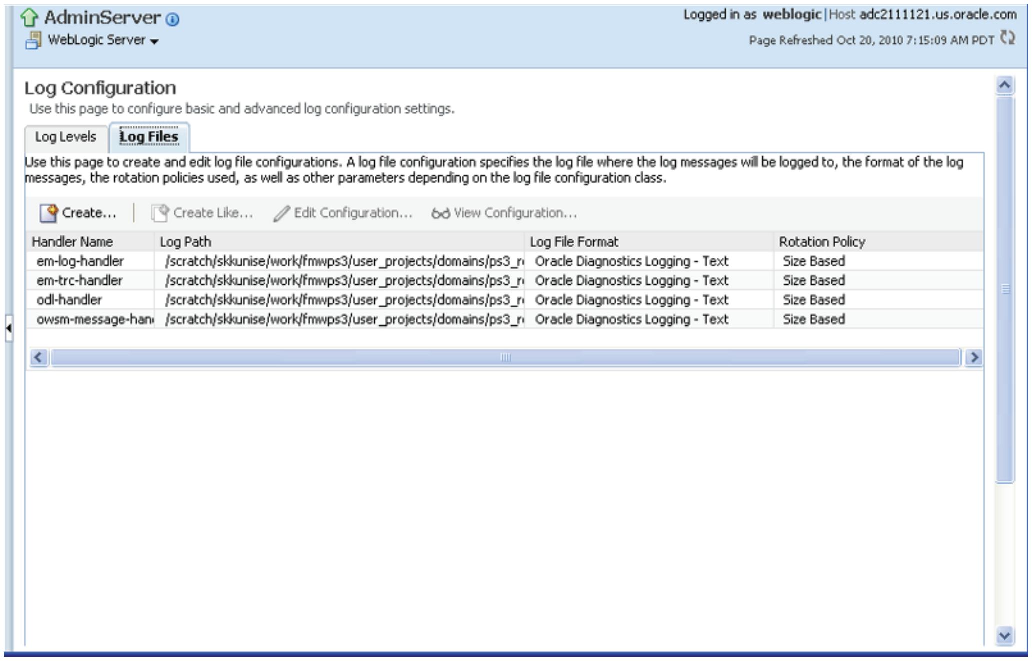
2. From the **WebLogic Sever** menu, select **Logs > Log Configuration**.

   The Log Configuration page is displayed.

3. Select the **Log Files** tab.

   The current list of log files is displayed, as shown in Figure 16–8. The Log Configuration page shows the currently configured log path, file format, and rotation policy.

Figure 16–8 Current Log Files



AdminServer
WebLogic Server

Logged in as weblogic | Host adc2111121.us.oracle.com
Page Refreshed Oct 20, 2010 7:15:09 AM PDT

Log Configuration

Use this page to configure basic and advanced log configuration settings.

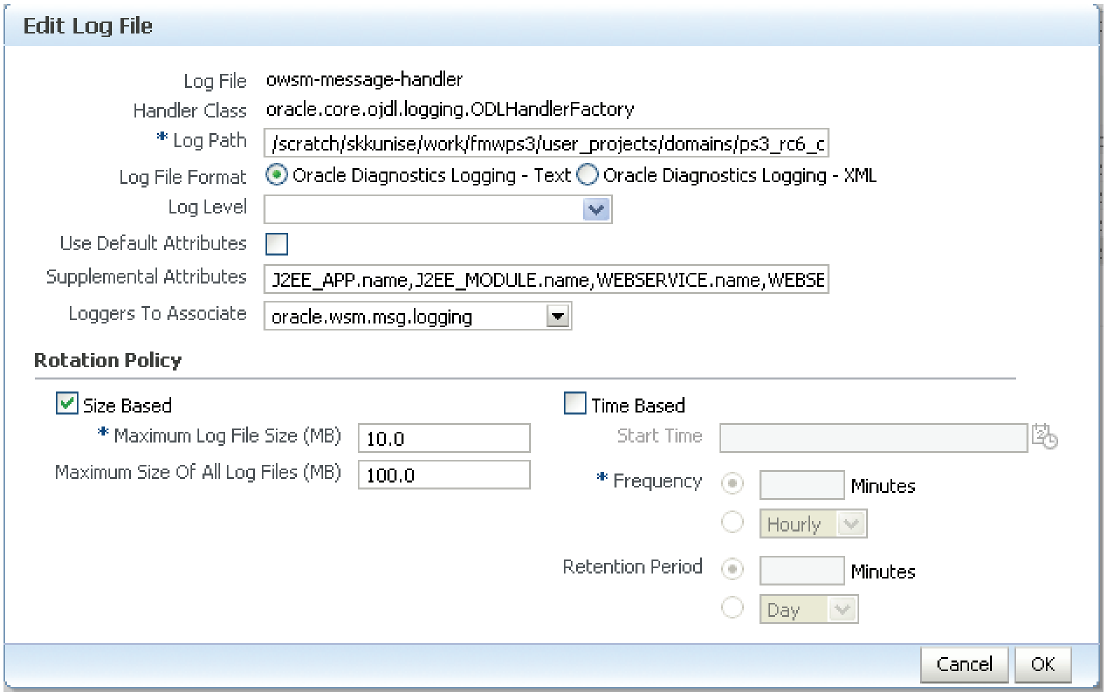
Use this page to create and edit log file configurations. A log file configuration specifies the log file where the log messages will be logged to, the format of the log messages, the rotation policies used, as well as other parameters depending on the log file configuration class.

Handler Name	Log Path	Log File Format	Rotation Policy
em-log-handler	/scratch/skkunise/work/fmwps3/user_projects/domains/ps3_r	Oracle Diagnostics Logging - Text	Size Based
em-trc-handler	/scratch/skkunise/work/fmwps3/user_projects/domains/ps3_r	Oracle Diagnostics Logging - Text	Size Based
odl-handler	/scratch/skkunise/work/fmwps3/user_projects/domains/ps3_r	Oracle Diagnostics Logging - Text	Size Based
owsm-message-han	/scratch/skkunise/work/fmwps3/user_projects/domains/ps3_r	Oracle Diagnostics Logging - Text	Size Based

4. If you wish to edit the log policy configuration, select the log file in the list and click **Edit Configuration . . .**.

   The Edit Log File page is displayed.

Figure 16–9 Edit Log File Page



Edit Log File

Log File	owsm-message-handler
Handler Class	oracle.core.ojdl.logging.ODLHandlerFactory
* Log Path	/scratch/skkunise/work/fmwps3/user_projects/domains/ps3_rc6_c
Log File Format	● Oracle Diagnostics Logging - Text ○ Oracle Diagnostics Logging - XML
Log Level	
Use Default Attributes	□
Supplemental Attributes	J2EE_APP.name,J2EE_MODULE.name,WEBSERVICE.name,WEBSE
Loggers To Associate	oracle.wsm.msg.logging

Rotation Policy

☑ Size Based

* Maximum Log File Size (MB)	10.0
Maximum Size Of All Log Files (MB)	100.0

□ Time Based

Start Time		
* Frequency	●	Minutes
	○	Hourly
Retention Period	●	Minutes
	○	Day

5. Edit the log file information, as required.

Table 16–4 Fields in Edit Log File Page

Field	Description
Log Path	Path to the log file. This field is required.
Log File Format	Format of the log file. Valid values are text or XML.
Log Level	Default log level for the logger. Select a log level from the list. Valid values include: ■ INCIDENT_ERROR:1 (SEVERE+100) ■ ERROR:1 (SEVERE) ■ WARNING:1 (WARNING) ■ NOTIFICATION:1 (INFO) ■ NOTIFICATION:16 (CONFIG) ■ TRACE:1 (FINE) ■ TRACE:16 (FINER) ■ TRACE:32 (FINEST)
Use Default Attributes	Flag that specifies whether to use default attributes for the logger.
Supplemental Attributes	Supplemental attributes required.
Loggers to Associate	Components to associate with the logger.
Rotation Policy	Specify whether you wish to rotate log files based on file size of length of time. For more information, see "Configuring Log File Rotation" in *Oracle Fusion Middleware Administrator's Guide*. If Size Based is selected as the rotational policy, Maximum Log Files Size is a required field. If Time Based is selected as the rotational policy, Frequency is a required field.

6. Click **OK** to edit the log file configuration.

17

Maintaining the Oracle WSM Repository

The following topics provide guidance for maintaining the Oracle WSM Repository:

17.1 About the Oracle WSM Repository

Oracle Web Services Manager (WSM) uses an MDS repository to store Oracle WSM metadata, such as policies, assertion templates, and policy usage data. The Oracle WSM Repository is available as a database (for production use) or as files in the file system (for development use in JDeveloper).

For a list of the databases that are supported for this release, see *Oracle Fusion Middleware Supported System Configurations*.

Within the Oracle WSM Repository, each policy has a URI that is evaluated to form a path in which to locate a particular XML document containing the policy. Oracle WSM does not use the MDS customization feature, so all policies are stored as complete documents. Although MDS supports the ability to store multiple versions of a given document, Oracle WSM only accesses the latest version during policy enforcement.

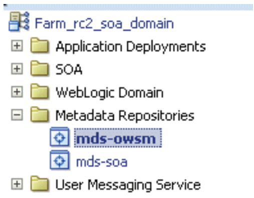
Details about managing the MDS repository are provided in "Managing the MDS Repository" in *Oracle Fusion Middleware Administrator's Guide*.

17.2 Registering an Oracle WSM Repository

Before you can deploy an application to an MDS Repository, such as the Oracle WSM Repository, you must register the repository with the Oracle WebLogic domain. To register an Oracle WSM Repository:

1. In the Navigator pane, expand **Metadata Repositories** and select **mds-owsm**, as shown in Figure 17–1.

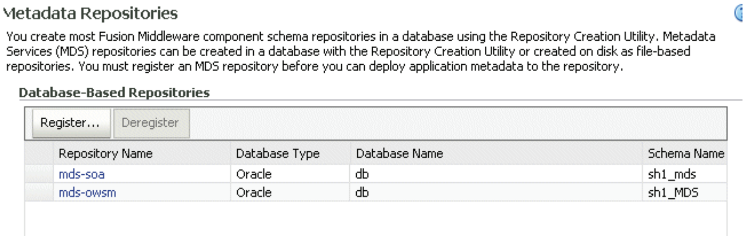
Figure 17–1 Metadata Repository in Navigation Pane



2. Select **Metadata Repository**, then **Administration**, then **Register/Deregister**.

   The Metadata Repositories page is displayed, as shown in Figure 17–2.

Figure 17–2 Registering an Oracle WSM Repository



Metadata Repositories

You create most Fusion Middleware component schema repositories in a database using the Repository Creation Utility. Metadata Services (MDS) repositories can be created in a database with the Repository Creation Utility or created on disk as file-based repositories. You must register an MDS repository before you can deploy application metadata to the repository.

Database-Based Repositories

Repository Name	Database Type	Database Name	Schema Name
mds-soa	Oracle	db	sh1_mds
mds-owsm	Oracle	db	sh1_MDS

3. Click **Register** and provide the required database connection and repository information to register the repository.

   Complete details for registering and managing a metadata repository are provided in "Managing the Metadata Repository" in the *Oracle Fusion Middleware Administrator's Guide*.

17.3 Understanding the Different Mechanisms for Importing and Exporting Policies

You can use Enterprise Manager Fusion Middleware Control or WebLogic Scripting Tool (WLST) commands to import and export policies to and from the Oracle WSM Repository.

Oracle Enterprise Manager Fusion Middleware Control provides the ability to selectively import and export one policy at a time. The procedures for importing and exporting policies using Fusion Middleware Control are described in the following sections:

- "Importing Web Service Policies" on page 7-7
- "Exporting Web Service Policies" on page 7-20

The WLST commands, `importRepository` and `exportRepository`, are provided to facilitate importing and exporting multiple Oracle WSM documents directly to and from the Oracle WSM Repository. For details about using these commands, see "Importing and Exporting Documents in the Repository" on page 17-3.

When you import or export policies using either of these mechanisms, the operation is routed through an instance of the Oracle WSM Policy Manager application. At run time, when a request for a policy is made, the Policy Manager guarantees that the latest policy is always provided. Therefore, the latest policies are always enforced.

Note: In earlier releases, the only WLST commands available to import and export polices were the `importMetadata` and `exportMetadata` MDS WLST commands. Oracle does not recommend using these commands for Oracle WSM documents because the operation is not routed through an instance of the Oracle WSM Policy Manager. Consequently, Oracle Web Services Manager may not be aware of the changes and may continue to enforce outdated policies. To ensure that only the latest polices are enforced, you must restart all the servers to which the Oracle WSM MDS repository is registered.

17.4 Importing and Exporting Documents in the Repository

You can import and export Oracle WSM documents to and from the Oracle WSM Repository using the `importRepository` and `exportRepository` WLST commands as described in the following sections:

- Exporting Documents from the Repository
- Importing Documents into the Repository

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

17.4.1 Exporting Documents from the Repository

To export documents from the repository to a supported ZIP archive file, use the `exportRepository` command.

```
exportRepository(archive,[documents=None],[expandReferences='false'])
```

Note the following:

- If the archive specified using the `archive` argument already exists, you can choose to merge the documents into the existing archive, overwrite the existing archive, or cancel the operation.
- Use the optional `documents` argument to specify the documents you want exported to the archive. If you do not specify this argument, then all assertion templates, intents, policies, and policy sets are exported. You can specify a list of the documents to be exported, or use a search expression to find specific documents in the repository. For example, to export a list of policies whose URI begins with either "oracle/wss10_" or "oracle/wss11_", enter the following:

```
wls:/jrfServer_
domain/serverConfig>exportRepository('/tmp/test2.zip',['policies:oracle/wss10_
%','policies:oracle/wss11_%'])

Exporting "/policies/oracle/wss10_message_protection_client_policy"
Exporting "/policies/oracle/wss10_message_protection_service_policy"
.
.
.
Exporting "/policies/oracle/wss11_x509_token_with_message_protection_client_
policy"
Exporting "/policies/oracle/wss11_x509_token_with_message_protection_service_
policy"
Successfully exported "43" documents.
```

- Use the optional `expandReferences` argument to expand the policy references during the export. The default is `false`. When no list of documents is provided, `expandReferences` is `true`.

  For example, to export active policy set documents and the policies they use:

  ```
  wls:/jrfServer_
  domain/serverConfig>exportRepository('/tmp/repository-active.jar',
  ['policysets:global/%'], true)

  Exporting "/policies/oracle/wsaddr_policy"
  Exporting "/policies/oracle/wss11_saml_or_username_token_with_message_
  protection_service_policy"
  Exporting "/policies/oracle/wss_username_token_service_policy"
  Exporting "/policysets/global/all-domains-default-web-service-policies"
  Exporting "/policysets/global/app-only-web-service-policies"
  Exporting "/policysets/global/migrate_example"
  Successfully exported "6" documents.
  ```

- If you modify a document in the repository, you can update it in the archive file. For example, if you modified a policy set named module-web-service-policies, you can update the policy set in the archive using the following command:

  ```
  wls:/jrfServer_
  domain/serverConfig>exportRepository('/tmp/repository-backup.jar',
  ['/policysets/global/module-web-service-policies'])
  ```

17.4.2 Importing Documents into the Repository

To import documents into the repository use the `importRepository` command.

```
importRepository(archive,[map=none],[generateMapFile='false'])
```

Note the following:

- The `archive` argument, which is required, specifies the path to the archive file that contains the list of documents to be imported. If a document being imported is a duplicate of the current version that already exists in the repository, then it will not be imported and a new version of the document is not created.
- Optionally, you can use the `map` argument to provide the location of a file that describes how to map physical information in a policy set, from the source environment to the target environment. For example, you can use the map file to ensure that the resource scope expression in a policy set is updated to match the target environment, such as `Domain("foo")=Domain("bar")` If you specify a map file and it does not exist, the operation fails and an error is displayed.
- You can set the optional `generateMapFile` argument to `true` to create a sample map file at the location specified by the `map` argument. No documents are imported when this argument is set to `true`. The default is `false`.

  After the map file is created you can edit it using any text editor. The map file contains the document names given in the archive file and their corresponding `attachTo` values. The `attachTo` value can be updated to correspond to the new environment. If a mapping update is not required for a document name, that entry may be either deleted or commented out using the `#` character.

Note: When importing documents into the repository, OWSM validates the `attachTo` values only. If a value is invalid, then the policy set is disabled. Other text in the map file is not validated.

For example, to generate a map file /tmp/mapfile.txt for the /tmp/repository-active.jar, enter the following command:

```
wls:/jrfServer_
domain/serverConfig>importRepository('/tmp/repository-active.jar',
'/tmp/mapfile.txt', true)

Successfully generated "Resource Scope Mappings" file at "/tmp/mapfile.txt"
```

To import the active policy set archive /tmp/repository-active.jar using the map file /tmp/mapfile.txt, enter the following:

```
wls:/jrfServer_domain/serverConfig>importRepository('/tmp/repository-active.jar',
'/tmp/mapfile.txt')

Importing "META-INF/policies/oracle/wsaddr_policy"
Importing "META-INF/policies/oracle/wss11_saml_or_username_token_with_message_
protection_service_policy"
Importing "META-INF/policies/oracle/wss_username_token_service_policy"
Importing "META-INF/policysets/global/all-domains-default-web-service-policies"
Importing "META-INF/policysets/global/app-only-web-service-policies"
Importing "META-INF/policysets/global/migrate_example"
Successfully imported "6" documents
```

17.5 Migrating Policies Between Application Environments

Policies can be migrated through the different stages of the application development and deployment cycles, such as from development to production. Oracle recommends using the `importRepository` and `exportRepository` commands for policy migration, as described in "Migrating Policies" on page 15-4.

17.5.1 Exporting Policies from the Oracle WSM Repository for Use in JDeveloper

In JDeveloper, you can add custom policies to the default policy store location at:

```
 C:\Documents and
Settings\user-dir\ApplicationData\JDeveloper\system11.1.1.2.x.x.x\DefaultD
omain\oracle\store\gmds
```

Within this directory, Oracle WSM policies files must be included using one of the following directory structures:

- Predefined Oracle WSM policies: `owsm/policies/oracle/`*`policy_file`*
- Custom user policies: `owsm/policies/`*`policy_file`*

When exporting policy files from the Oracle WSM Repository for use in JDeveloper, this directory structure is not maintained. You must ensure that when adding the exported policy to the JDeveloper environment that you use the required directory structure noted above. Otherwise, the policies will not be available in the JDeveloper environment.

17.6 Patching Policies in the Repository

You can patch the Oracle WSM Repository using either Fusion Middleware Control or the WLST commands, as described in "Understanding the Different Mechanisms for Importing and Exporting Policies" on page 17-2. When you create or update a policy, there are two possible scenarios to consider when you patch the repository:

- You create a new policy or update an existing policy that uses a new policy URI. In this scenario, the patching of the repository acts as if a new file was added to the installation and, as a result, only impacts the components that expect to use the new policy. Once loaded, the policy is available to all applications. Generally speaking, using a new policy URI is the preferred model as policies are typically named to convey the behavior they represent.
- You create a new policy or update an existing policy that uses an existing policy URI. In this scenario, the patching of the repository acts as if an existing file was overwritten with a new version and, therefore, impacts all components that are using the existing policy. Once loaded, all applications will use the new version of the policy. Reusing an existing URI is typically only done to make minor modifications to the behavior of a policy. Note that if you use WLST commands to patch the repository, you need to restart the server to ensure that the latest version of the policy is enforced. You do not need to restart if you use Fusion Middleware Control.

17.7 Backing Up and Restoring the Oracle WSM Repository

Use the `exportRepository` and `importRepository` WLST commands to back up and restore the Oracle WSM Repository. For more information about these commands, see "Importing and Exporting Documents in the Repository" on page 17-3.

For example, to backup all the Oracle WSM artifacts in the repository, enter the following command:

```
wls:/jrfServer_domain/serverConfig>exportRepository('/tmp/repository-backup.jar')

Exporting "/assertiontemplates/oracle/binding_authorization_template"
Exporting "/assertiontemplates/oracle/binding_permission_authorization_template"
.
.
.
Exporting "/policies/oracle/binding_authorization_denyall_policy"
Exporting "/policies/oracle/binding_authorization_permitall_policy"
.
.
.
Exporting "/policysets/global/all-domains-default-web-service-policies"
Exporting "/policysets/global/app-only-web-service-policies"
Successfully exported "170" documents.
```

To restore the repository from the backup, use the `importRepository` command to import all the Oracle WSM Repository artifacts.

For example, to restore the repository using the backup file created in the previous example, enter the following command:

```
wls:/jrfServer_domain/serverConfig>importRepository('/tmp/repository-backup.jar')

Importing "META-INF/assertiontemplates/oracle/binding_authorization_template"
Importing "META-INF/assertiontemplates/oracle/binding_permission_authorization_
template"
```

```
.
.
.
Importing "META-INF/policies/oracle/binding_authorization_denyall_policy"
Importing "META-INF/policies/oracle/binding_authorization_permitall_policy"
.
.
.
Importing "META-INF/policysets/global/all-domains-default-web-service-policies"
Importing "META-INF/policysets/global/app-only-web-service-policies"
Successfully imported "170" documents.
```

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WebLogic Scripting Tool Command Reference*.

17.8 Upgrading the Oracle WSM Policies in the Repository

Both predefined and custom Oracle WSM policies are stored in the Oracle WSM Repository. In subsequent releases, the predefined policies may be discontinued, changed, or new predefined policies may be provided.

After you install a Fusion Middleware patch set, the repository is automatically upgraded as part of the server startup process. Any predefined policies that have not been customized for your environment are replaced, and any new policies are automatically added. Note, however, that predefined policies that have been customized and user-created custom policies in the repository are not replaced. If desired, you can refresh the repository for these policies also, as described in "Rebuilding the Oracle WSM Repository" on page 17-7.

Note: You should back up your existing policies to a safe location before deleting any policies. In the event you have any issues with the new policies, you can import the existing policies from the backup.

For details about patching your Oracle Fusion Middleware installation, see *Oracle Fusion Middleware Patching Guide*.

17.9 Rebuilding the Oracle WSM Repository

In some circumstances, it may be desirable to rebuild the entire Oracle WSM Repository, including restoring the original predefined policies and assertion templates. For example, when starting a new project in a test environment it may be useful to reset the repository contents to their original state.

To rebuild the Oracle WSM Repository, perform the following steps:

1. Connect to the Administration Server instance of the WebLogic Server domain to which the repository is registered. For instructions, see "Accessing the Web Services Custom WLST Commands" on page 1-6.

   Note: You should back up your existing policies to a safe location before deleting any policies or rebuilding the repository. In the event you have any issues with the new policies, you can import the existing policies from the backup.

2. Use the `resetWSMPolicyRepository(true)` command to delete all the documents from the Oracle WSM Repository and repopulate it with the set of predefined policies and assertion templates that were installed with the software. This is the preferred method.

   For more information about the `resetWSMPolicyRepository` WLST command, see "Oracle WSM Repository Management Commands" in *WebLogic Scripting Tool Command Reference*.

> **Note:** Before you delete a policy, Oracle recommends that you verify that the policy is not attached to any policy subjects.

Part IV

WebLogic Web Service Administration

Part IV contains the following chapter:

18

Securing and Administering WebLogic Web Services

This chapter describes how to secure and administer WebLogic Web services, including the following sections:

- Steps to Secure and Administer WebLogic Web Services
- Attaching Policies to WebLogic Web Services and Clients

18.1 Steps to Secure and Administer WebLogic Web Services

Table 18–1 summarizes the steps required to administer and secure WebLogic Web services. For information about developing WebLogic Web services, see *Getting Started With JAX-WS Web Services for Oracle WebLogic Server*.

Table 18–1 Steps to Administer and Secure WebLogic Web Services

#	Step	Description
1	Deploy and administer the WebLogic Web service.	Use the Oracle WebLogic Server Administration Console to perform the following deployment and administration tasks: ■ Deploy a WebLogic Web service and view deployed services. ■ Start and stop a WebLogic Web service. ■ View the WebLogic Web service configuration. ■ Delete a WebLogic Web service. ■ View the SOAP message handlers. ■ View the WSDL. For more information, see "Web Services" in the *Oracle WebLogic Server Administration Console Help*.
2	Attach the security and management policies to your WebLogic Web services and clients.	You can attach two types of policies to WebLogic Web services and clients at design and deployment time: Oracle WSM and WebLogic Web service policies. You can use Oracle Enterprise Manager Fusion Middleware Control to attach Oracle WSM security policies to WebLogic Java EE Web services and clients. For details, see "Attaching Policies to WebLogic Web Services and Clients" on page 18-2.
3	Test the WebLogic Web services.	See "Testing Web Services" on page 12-1.
4	Monitor the performance of WebLogic Web services.	See "Monitoring the Performance of Web Services" on page 13-1.

18.2 Attaching Policies to WebLogic Web Services and Clients

In Oracle Fusion Middleware 11*g* Release 1 (11.1.1.9), you can provide security and management policy enforcement of WebLogic Web services using one of the following policy types: *Oracle WSM* or *WebLogic Web service*.

The following table describes each policy type.

Table 18–2 Policy Types Supported by WebLogic Web Services

Type	Description
Oracle Web Services Manager (WSM) Policy	Provided by the Oracle WSM. For more information about Oracle WSM and the predefined policies, see "Understanding Oracle WSM Policy Framework" on page 3-1. You can attach Oracle WSM policies to WebLogic JAX-WS Web services and clients.
WebLogic Web Service Policy	Provided by Oracle WebLogic Server. For more information about the WebLogic Web service policies, see *Securing WebLogic Web Services for Oracle WebLogic Server*. A subset of WebLogic Web service policies interoperate with Oracle WSM policies. For more information, see "Interoperability with Oracle WebLogic Server 11g Web Service Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.

Note: It is recommended that you use Oracle WSM policies whenever possible. You cannot mix your use of Oracle WSM and WebLogic Web service policies.

The following sections describe how to attach each type of policy to WebLogic Web services and clients.

- Attaching Oracle WSM Policies to WebLogic Web Services
- Attaching Oracle WSM Policies to WebLogic Web Service Clients
- Attaching WebLogic Web Service Policies to WebLogic Web Services
- Attaching WebLogic Web Service Policies to WebLogic Web Service Clients

18.2.1 Attaching Oracle WSM Policies to WebLogic Web Services

You attach Oracle WSM policies to WebLogic Web services at design time and after the Web service has been deployed.

- At design time, use the `weblogic.wsee.jws.jaxws.owsm.SecurityPolicy` and `weblogic.wsee.jws.jaxws.owsm.SecurityPolicies` JWS annotations in your JWS file to associate policy files with your Web service. You can associate any number of policy files with a Web service, although it is up to you to ensure that the assertions do not contradict each other. You can specify a policy file at the class level of your JWS file. For more information, see the following sections:
  - "Using Oracle Web Services Manager Security Policies" in *Securing WebLogic Web Services for Oracle WebLogic Server*.
  - "Using Policies with Web Services" in "Developing with Web Services" in the Oracle JDeveloper online help.
- After the Web service has been deployed, you can use the Oracle WebLogic Server Administration Console or Oracle Enterprise Manager Fusion Middleware Control to attach Oracle WSM policies to WebLogic Web services. For more information about attaching policies using the WebLogic Server Administration Console, see "Attach a WS-Policy file to a Web Service" in the *Oracle WebLogic Server*

Administration Console Help. For more information about attaching policies using Fusion Middleware Control, see Chapter 8, "Attaching Policies to Web Services."

18.2.2 Attaching Oracle WSM Policies to WebLogic Web Service Clients

Oracle recommends that you use Oracle Enterprise Manager Fusion Middleware Control to attach Oracle WSM policies to a Web service client post-deployment. For more information, see "Attaching Policies to Java EE Web Service Clients" on page 8-13. If you attach Oracle WSM policies programmatically at development time, you will not be able to modify or delete the policies using Fusion Middleware Control after the client application is deployed.

18.2.3 Attaching WebLogic Web Service Policies to WebLogic Web Services

You attach policies to WebLogic Web services at both design time and after the Web service has been deployed.

- At design time, use the `weblogic.jws.Policy` and `weblogic.jws.Policies` JWS annotations in your JWS file to associate policy files with your Web service. You can associate any number of policy files with a Web service, although it is up to you to ensure that the assertions do not contradict each other. You can specify a policy file at the class level of your JWS file. For more information, see the following sections:
  - *Securing WebLogic Web Services for Oracle WebLogic Server*.
  - "Using Policies with Web Services" in "Developing with Web Services" in the Oracle JDeveloper online help.
- After the Web service has been deployed, use the Oracle WebLogic Server Administration Console to attach WebLogic Web service policies to WebLogic Web services. For more information, see "Attach a WS-Policy file to a Web Service" in the *Oracle WebLogic Server Administration Console Help*.

18.2.4 Attaching WebLogic Web Service Policies to WebLogic Web Service Clients

You attach policies to WebLogic Web service clients at design time, using JAX-WS Stubs. For more information, see "Using a Client-side Security Policy File" in *Securing WebLogic Web Services for Oracle WebLogic Server*.

Part V

Reference

Part V contains the following chapters:

A

Web Service Security Standards

Note: This appendix summarizes the security standards for Oracle Infrastructure Web Services. For a complete list of the versions supported with links to the specifications, see "Supported Standards" in *Developer's Guide for Oracle Infrastructure Web Services*.

For a description of standards for WebLogic Web services, see "Features and Standards Supported by WebLogic Web Services" in *Introducing WebLogic Web Services for Oracle WebLogic Server*

Security standards are implemented in non-XML frameworks at the transport level, and in XML frameworks at the application level.

The following sections describe the standards that are key to providing secure and manageable SOA environments at both the transport and application levels.

- Web Services Interoperability Organization—Basic Security Profile
- Transport Layer Security—SSL
- XML Encryption (Confidentiality)
- XML Signature (Integrity, Authenticity)
- WS-Security
- WS-Security Tokens
- WS-Policy
- WS-SecurityPolicy
- Web Services Addressing (WS-Addressing)
- WS-Trust
- WS-ReliableMessaging

See Also: For a complete list of standards supported by Oracle WebLogic Web services, see "Features and Standards Supported by WebLogic Web Services" in *Introducing WebLogic Web Services for Oracle WebLogic Server*.

A.1 Web Services Interoperability Organization—Basic Security Profile

Oracle considers interoperability of Web services platforms to be more important than providing support for all possible edge cases of the Web services specifications. Oracle complies with the following specification from the Web Services Interoperability Organization and considers it to be the baseline for Web services interoperability:

- *Basic Security Profile 1.0*: `http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html`

A.2 Transport Layer Security—SSL

Secure Sockets Layer (SSL), also known as Transport Layer Security (TLS), is the most widely used transport-layer data-communication protocol. SSL provides the following:

- Authentication—communication is established between two trusted parties.
- Message confidentiality—data exchanged is encrypted.
- Message integrity—data is checked for corruption.
- Secure key exchange between client and server

SSL can be used in three modes:

- No authentication: Neither the client nor the server authenticates itself to the other. No certificates are sent or exchanged. In this case, only confidentiality (encryption/decryption) is used.
- One-way authentication (or server authentication): Only the server authenticates itself to the client. The server sends the client a certificate verifying that the server is authentic. This is typically the approach used for Internet transactions such as online banking.
- Two-way authentication (or bilateral authentication): Both client and server authenticate themselves to each other by sending certificates to each other. This approach is necessary to prevent attacks from occurring between a proxy and a Web service endpoint.

SSL uses a combination of secret-key and public-key cryptography to secure communications. SSL traffic uses secret keys for encryption and decryption, and the exchange of public keys is used for mutual authentication of the parties involved in the communication.

A.3 XML Encryption (Confidentiality)

The XML encryption specification describes a process for encrypting data and representing the result in XML. Specifically, XML encryption defines:

- How digital content is encrypted and decrypted.
- How the encryption key information is passed to a recipient.
- How encrypted data is identified to facilitate encryption.

An XML document may be encrypted as a whole or in part.

Example A–1 illustrates credit card data represented in XML.

Example A–1 XML Representation of Credit Card Data

```
<PaymentInfo xmlns="http://www.example.com/payment">
  <CreditCard>
```

```
    <Name>John Smith</Name>
    <CreditCardNumber>4019 2445 0277 5567</NCreditCardNumber>
    <Limit>5000</Limit>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Example A–2 illustrates the same XML snippet with the credit card number encrypted and represented by a cipher value.

Example A–2 XML Representation of Encrypted Credit Card Data

```
<PaymentInfo xmlns='http://www.example.com/payment">
  <CreditCard>
    <Name>John Smith</Name>
    <CreditcardNumber>
      <EncryptedData xmlns="http://www..." Type="http://www...">
        <CipherData>
          <CipherValue>A23B4...5C56</CipherValue>
        </CipherData>
      </EncryptedData>
    <Limit>5000</Limit>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

See Also:

For more information about XML encryption, see "XML Encryption Syntax and Processing" specification at:

http://www.w3.org/TR/xmlenc-core

A.4 XML Signature (Integrity, Authenticity)

The XML Signature specification describes signature processing rules and syntax. XML Signature binds the sender's identity (or "signing entity") to an XML document. The document is signed using the sender's private key; the signature is verified using the sender's public key.

Signing and signature verification can be done using asymmetric or symmetric keys. XML Signature also ensures non-repudiation of the signing entity, that is, it provides proof that messages have not been altered since they were signed.

A signature can apply to a whole document or just part of a document, as shown in the following example.

Example A–3 XML Representation of Signed Data

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<!-- The signedInfo element allows us to sign any portion of a
 document -->
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www..."/>
    <SignatureMethod Algorithm="http://www..."/>
    <Reference URI="#Body">
      <DigestMethod Algorithm="http://www..."/>
      <DigestValue>o+jtqlieRtF6DrUb...X8O9M/CmySg</DigestValue>
    </Reference>
  </SignedInfo>
  <!-- Following is the result of running the algorithm over the
  document. If changes are made to the document, the SignatureValue is
  changed. The security application verifies the SignatureValue,
  extracts the X.509 cert and uses it to authenticate the user -->
  <SignatureValue>oa+ttbsvSFi...EtRD2oNC5</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <!-- Following is the public key that matches the private key
      that signs the document -->
      <RSAKeyValue>
        <Modulus>5TT/oolzTiP++Ls6GLQUM8xoFFrAlZQ...</Modulus>
        <Exponent>EQ==</Exponent>
      </RSAKeyValue>
    </KeyValue>
    <!-- Following is the certificate -->
    <X509Data>
      <X509Certificate>wDCCAXqgAwIBAgI...</X509Certificate>
    </X509Data>
  </KeyInfo>
</Signature>
```

See Also:

For more information about XML Signature, see the "XML Signature Syntax and Processing" specification at:

http://www.w3.org/TR/xmldsig-core

A.5 WS-Security

Web Services Security (WS-Security) specifies SOAP security extensions that provide confidentiality using XML Encryption and data integrity using XML Signature. WS-Security also includes profiles that specify how to insert different types of binary and XML security tokens in WS-Security headers for authentication and authorization purposes. WS-Security token profiles are described in the following sections

See Also:

For more information about WS-Security and its specification, see:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

A.6 WS-Security Tokens

Web services security supports the following security tokens:

- Username—defines how a Web service consumer can supply a username as a credential for authentication). For more information, see "Username" on page A-5
- X.509 certificate—a signed data structure designed to send a public key to a receiving party. For more information, see "X.509 Certificate" on page A-5
- Kerberos ticket—a binary authentication and session token. For more information, see "Kerberos Token" on page A-5
- Security Assertion Markup Language (SAML) assertion—shares security information over the Internet through XML documents. For more information, see "SAML Token" on page A-6

A.6.1 Username

The username token carries basic authentication information. The `username-token` element propagates username and password information to authenticate the message.

See Also:

For more information about the username token profile, see:

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

A.6.2 X.509 Certificate

An X.509 digital certificate is a signed data structure designed to send a public key to a receiving party. A certificate includes standard fields such as certificate ID, issuer's Distinguished Name (DN), validity period, owner's DN, owner's public key, and so on.

Certificates are issued by certificate authorities (CA). A CA verifies an entity's identity and grants a certificate, signing it with the CA's private key. The CA publishes its own certificate which includes its public key.

Each network entity has a list of the certificates of the CAs it trusts. Before communicating with another entity, a given entity uses this list to verify that the signature of the other entity's certificate is from a trusted CA.

See Also:

For more information about the X.509 certificate token profile, see:

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf

A.6.3 Kerberos Token

Kerberos token is a cross-platform authentication and single sign-on system. The Kerberos protocol provides mutual authentication between two entities relying on a shared secret (symmetric keys). Kerberos uses the following terminology:

- A Principal is an identity for a user (i.e., a user is assigned a principal), or an identity for an application offering Kerberos services.
- A Realm is a Kerberos server environment; a Kerberos realm can be a domain name such as EXAMPLE.COM (by convention expressed in uppercase).

See Also:

For more information about the Kerberos token profile 1.1, see:

http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf

Kerberos involves a client, a server, and a trusted party to mediate between them called the Key Distribution Center (KDC). Each Kerberos realm has at least one KDC. KDCs come in different packages based on the operating platform used (for example, on Microsoft Windows, the KDC is a domain service). The Kerberos Token profile of WS-Security allows business partners to use Kerberos tokens in service-oriented architectures.

A.6.4 SAML Token

The Security Assertion Markup Language (SAML) is an open framework for sharing security information over the Internet through XML documents. SAML was designed to address the following:

- Limitations of web browser cookies to a single domain: SAML provides a standard way to transfer cookies across multiple Internet domains.
- Proprietary web single sign-on (SSO): SAML provides a standard way to implement SSO within a single domain or across multiple domains. This functionality is provided by the Oracle Identity Federation product.
- Federation: SAML facilitates identity management (e.g., account linking when a single user is known to multiple web sites under different identities), also supported by Oracle Identity Federation.
- Web Services Security: SAML provides a standard security token (a SAML assertion) that can be used with standard web services security frameworks (e.g., WS-Security) – This is the use of SAML that is particularly relevant to web services security, fully supported by Oracle WSM.
- Identity propagation: SAML provides a standard way to represent a security token that can be passed across the multiple steps of a business process or transaction, from browser to portal to networks of web services, also a feature supported by Oracle WSM.

The SAML framework includes 4 parts:

- Assertions: How you define authentication and authorization information.
- Protocols: How you ask (SAML Request) and get (SAML Response) the assertions you need.
- Bindings: How SAML Protocols ride on industry-standard transport (e.g., HTTP) and messaging frameworks (e.g., SOAP).
- Profiles: How SAML Protocols and Bindings combine to support specific use cases.

In the context of WS-Security, only SAML assertions are used. The protocols and bindings are provided by the WS-Security framework. SAML is widely adopted by the industry, both for browser-based federation and federation enabled by web services flows.

SAML assertions are very popular security tokens within WS-Security because they are very expressive and can help prevent man-in-the-middle and replay attacks.

Typically, a SAML assertion makes statements about a principal (a user or an application). All SAML assertions include the following common information:

- Issuer ID and issuance timestamp
- Assertion ID
- Subject
- Name
- Optional subject confirmation (for example, a public key)
- Optional conditions (under which an assertion is valid)
- Optional advice (on how an assertion was made)

SAML assertions can include three types of statements:

- Authentication statement: issued by an authentication authority upon successful authentication of a subject. It asserts that Subject S was authenticated by Means M at Time T.
- Attribute statement: issued by an attribute authority, based on policies. It asserts that Subject S is associated with Attributes A, B, etc. with values a, b, and so on.
- Authorization decision statement (deprecated in SAML 2.0, now supported by XACML): issued by an authorization authority which decides whether to grant the request by Subject S, for Action A (e.g., read, write, etc.), to Resource R (e.g., a file, an application, a Web service), given Evidence E.

SAML assertions can be embedded (i.e., a SAML assertion can contain another SAML assertion). SAML assertions can be signed (using XML Signature) and/or encrypted (using XML Encryption).

See Also:

For more information about the SAML token profile, see:

`http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf`

A.7 WS-Policy

Together with WS-Security, WS-Policy is another key industry standard for Oracle Fusion Middleware security.

A Web service provider may define conditions (or policies) under which a service is to be provided. The WS-Policy framework enables one to specify policy information that can be processed by web services applications, such as Oracle WSM.

A policy is expressed as one or more policy assertions representing a Web service's capabilities or requirements. For example, a policy assertion may stipulate that a request to a Web service be encrypted. Likewise, a policy assertion can define the maximum message size that a Web service can accept.

WS-Policy expressions are associated with various web services components using the WS-PolicyAttachment specification. WS-Policy information can be embedded in a WSDL file, thus making it easy to expose Web service policies through a UDDI registry.

A.8 WS-SecurityPolicy

WS-SecurityPolicy is part of the Web Services Secure Exchange (WS-SX) set of specifications hosted by OASIS (in addition to WS-SecurityPolicy, the WS-SX technical committee defines two other sets of specifications: WS-Trust and WS-SecureConversation, described later in this chapter).

WS-SecurityPolicy defines a set of security policy assertions used in the context of the WS-Policy framework. WS-SecurityPolicy assertions describe how messages are secured on a communication path. Oracle has contributed to the OASIS WS-SX technical committee several practical security scenarios (a subset of which is provided by Oracle WSM 11*g*). Each security scenario describes WS-SecurityPolicy policy expressions.

WS-SecurityPolicy *scenarios* describe examples of how to set up WS-SecurityPolicy policies for several security token types described in the WS-Security specification (supporting both WS-Security 1.0 and 1.1). The subset of the WS-SecurityPolicy scenarios supported by Oracle WSM 11*g* represents the most common customer use cases. Each scenario has been tested in multiple-vendor WS-Security environments.

To illustrate WS-SecurityPolicy, let's use a scenario supported by Oracle WSM: UsernameToken with plain text password. As mentioned earlier, Username token is one of the security tokens specified by WS-Security. This specific scenario uses a policy that says that a requester must send a password in a Username token to a recipient who has authority to validate that token. The password is a default requirement for the WS-Security Username Token Profile 1.1.

This scenario is only recommended when confidentiality of the password is not an issue, such as a pre-production test scenario with dummy passwords.

Example A–4   Example of WS-SecurityPolicy

```
<wsp:Policy>
  <sp:SupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken/>
    </wsp:Policy>
  </sp:SupportingTokens>
</wsp:Policy>
```

An example of a message that conforms to the above stated policy is shown below.

Example A–5   Example of Message Conforming to WS-SecurityPolicy

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="...">
  <soap:Header>
    <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">
      <wsse:UsernameToken>
        <wsse:Username>Marc</wsse:Username>
        <wsse:Password Type="http://docs.oasis open.org...>
           XYZ
        </wsse:Password>
        <wsse:Nonce EncodingType="...#Base64Binary">qB...</wsse:Nonce>
        <wsu:Created>2008-01-02T00:01:03Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <Oracle xmlns=http://xmlsoap.org/Oracle>
      <text>EchoString</text>
    </Oracle>
  </soap:Body>
</soap:Envelope>
```

The example above contains a <Nonce> element and a <Created> timestamp, which, while optional, are recommended to improve security of requests against replay and other attacks. A nonce is a randomly generated (unique) number. The timestamp can be used to define the amount of time the security token is valid.

A.9 Web Services Addressing (WS-Addressing)

SOAP does not provide a standard way to specify where a message is going or how responses or faults are returned. WS-Addressing provides an XML framework for identifying web services endpoints and for securing end-to-end endpoint identification in messages.

A Web service endpoint is a resource (such as an application or a processor) to which web services messages are sent.

The following is an example using WS-Addressing (`wsa` is the namespace for WSAddressing):

Example A–6 Example of WS-Addressing

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
   <S:Header>
      <wsa:MessageID>http://example.com/xyz-abcd-123</wsa:MessageID>
      <wsa:ReplyTo>
         <wsa:Address>http://example.myClient1</wsa:Address>
      </wsa:ReplyTo>
```

WS-Addressing is transport-independent; that is, the request may be over JMS and the response over HTTP. WS-Addressing is used with other WS-* specifications, such as WS-Policy.

A.10 WS-Trust

The WS-Trust 1.3 specification (`http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.html`) defines extensions to WS-Security that provide a framework for requesting and issuing security tokens, and to broker trust relationships. WS-Trust extensions provide methods for issuing, renewing, and validating security tokens.

To secure communication between a Web service client and a Web service, the two parties must exchange security credentials. As defined in the WS-Trust specification, these credentials can be obtained from a trusted SecurityTokenService (STS), which acts as trust broker. That is, the STS must be trusted by both the Web service client and the Web service in order to provide interoperable security tokens.

A.11 WS-ReliableMessaging

WS-ReliableMessaging (WS-RM) defines a framework for identifying and managing the reliable delivery of messages between Web services endpoints. WS-RM is predicated on the SOAP messaging structure (SOAP binding) and relies on WS-Security, WS-Policy, and WS-Addressing to provide reliable messaging.

WS-RM defines a reliable messaging (RM) source (the party that sends the message) and an RM destination (the party that receives the message). WS-RM mandates prerequisites, for example, trust between endpoints must be established, and the message and endpoints must be formally identified (this is achieved through the use of the complementary WS-* specifications mentioned earlier).

WS-RM Policy defines a policy assertion that leverages the WS-Policy framework in order to enable an RM destination and an RM source to describe their requirements for a given sequence.

B

Predefined Policies

This appendix summarizes the predefined policies and contains the following sections:

- Security Policies
- WS-Addressing Policies
- MTOM Attachment Policies
- Reliable Messaging Policies
- Management Policies
- No Behavior Policies

Oracle has been instrumental in contributing to emerging standards, in particular the specifications hosted by the OASIS Web Services Secure Exchange technical committee. Oracle has contributed to the OASIS WS-SX technical committee several practical security scenarios, a subset of which are implemented in the predefined policies.

> **Note:** For information about WebLogic Web service policies, see *Securing WebLogic Web Services for Oracle WebLogic Server*.

B.1 Security Policies

The following sections describe the security policies.

- Authentication Only Policies
- Message Protection Only Policies
- Message Protection and Authentication Policies
- WS-Trust Policies
- Authorization Only Policies
- Authorization Policies—Oracle Entitlements Server

B.1.1 Authentication Only Policies

The following authentication only policies are provided for SOAP and RESTful Web services.

Table B–1 summarizes the security policies that enforce authentication only and can be attached to both SOAP and RESTful Web services.

Table B–1 Authentication Only Policies—SOAP and RESTful Web Services

Client Policy	Service Policy	Authentication Transport
oracle/http_jwt_token_client_policy	oracle/http_jwt_token_service_policy	Yes
oracle/http_jwt_token_identity_switch_client_policy	oracle/http_jwt_token_service_policy	Yes
N/A	oracle/http_oam_token_service_policy	Yes
oracle/http_oauth2_token_client_policy	oracle/http_jwt_token_service_policy, or oracle/multi_token_rest_service_policy	
http_oauth2_token_opc_oauth2_client_policy	Reserved for use with Oracle Cloud.	
oracle/oauth2_config_client_policy	N/A	
oracle/http_saml20_token_bearer_client_policy	oracle/http_saml20_token_bearer_service_policy	Yes
Attach one of the following: - oracle/wss_http_token_client_policy - oracle/http_saml20_token_bearer_client_policy - oracle/http_jwt_token_identity_switch_client_policy - oracle/http_jwt_token_client_policy - Policy generated using SPNEGO assertion, as described in "oracle/http_spnego_token_client_template" on page C-24 To support HTTP OAM security, you must configure OAM Webgate to intercept the request. For more information, see "oracle/multi_token_rest_service_policy" on page B-5.	oracle/multi_token_rest_service_policy	Yes

Table B–2 summarizes the security policies that enforce authentication only for SOAP Web services and indicates whether the token is inserted at the transport layer or SOAP header.

Table B–2 Authentication Only Policies—SOAP Web Services Only

Client Policy	Service Policy	Authentication Transport	Authentication SOAP
oracle/wss_http_token_client_policy	oracle/wss_http_token_service_policy	Yes	No
oracle/wss_username_token_client_policy	oracle/wss_username_token_service_policy	No	Yes
oracle/wss10_saml_token_client_policy	oracle/wss10_saml_token_service_policy	No	Yes
oracle/wss10_saml20_token_client_policy	oracle/wss10_saml20_token_service_policy	No	Yes
oracle/wss11_kerberos_token_client_policy	oracle/wss11_kerberos_token_service_policy	No	Yes

B.1.1.1 oracle/http_jwt_token_client_policy

This policy includes a JWT token in the HTTP header. The JWT token is created automatically. The issuer name and subject name are provided either programmatically or declaratively through the policy. You can specify the audience restriction condition for this policy.

This policy can be enforced on any HTTP-based client endpoint.

This policy contains the following policy assertion: oracle/http_jwt_token_client_template. See "oracle/http_jwt_token_client_template" on page C-4 for more information about the assertion.

For information about configuring the policy, see "oracle/http_jwt_token_client_policy" on page 11-7.

B.1.1.2 oracle/http_jwt_token_identity_switch_client_policy

Performs dynamic identity switching by propagating a different identity than the one based on the authenticated subject. This policy includes a JSON Web Token (JWT) in the HTTP header. The JWT token is created automatically. The issuer name and subject name are provided either programmatically or declaratively through the policy. You can specify the audience restriction condition for this policy.

This policy can be enforced on any HTTP-based, SOAP, or REST client endpoint.

This policy contains the following policy assertion: oracle/http_jwt_token_client_template. See "oracle/http_jwt_token_client_template" on page C-4 for more information about the assertion.

For information about configuring the policy, see "oracle/http_jwt_token_identity_switch_client_policy" on page 11-8.

B.1.1.3 oracle/http_jwt_token_service_policy

This policy authenticates users using the username provided in the JWT token in the HTTP header.

This policy can be applied to any HTTP-based endpoint.

This policy contains the following policy assertion: oracle/http_jwt_token_service_template. See "oracle/http_jwt_token_service_template" on page C-8 for more information about the assertion.

For information about configuring the policy, see "oracle/http_jwt_token_client_policy" on page 11-7.

B.1.1.4 oracle/http_oam_token_service_policy

This policy verifies that the OAM agent has authenticated the user and has established an identity. This policy can be enforced on any HTTP-based endpoint.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_oam_token_service_template. See "oracle/http_oam_token_service_template" on page C-10 for more information about the assertion.

For more information about configuring the policy, see "oracle/http_oam_token_service_policy" on page 11-10.

B.1.1.5 oracle/http_oauth2_token_client_policy

This policy includes the OAuth2 access token in the HTTP header. The access token (AT) is obtained from the Mobile & Social OAuth2 Server. You can attach this policy to any HTTP-based client.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_oauth2_token_client_template. See "oracle/http_oauth2_token_client_template" on page C-11 for more information about the assertion.

This policy must always be attached with oracle/oauth2_config_client_policy.

For more information about configuring the policy, see "oracle/http_oauth2_token_client_policy" on page 11-11.

B.1.1.6 http_oauth2_token_opc_oauth2_client_policy

This policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the Mobile & Social OAuth2 Server. The property `oracle.oauth2.service` is set to true by default, which ensures that the client ID is used as the issuer for the user and client JWT tokens for the OAuth2 server. If `scope` has no value, (the default), the protocol, host and port (if available) are obtained from the service URL and used. This policy can be attached to any HTTP-based, SOAP or REST client.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_oauth2_token_client_template. See "oracle/http_oauth2_token_client_template" on page C-11 for more information about the assertion.

This policy must always be attached with oracle/oauth2_config_client_policy.

For more information about configuring the policy, see "oracle/http_oauth2_token_opc_oauth2_client_policy" on page 11-13.

B.1.1.7 oracle/oauth2_config_client_policy

This policy provides OAuth2 information on the client side. This information is used to invoke the Mobile and Social OAuth2 server for token exchange.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/oauth2_config_client_template. See "oracle/oauth2_config_client_template" on page C-18 for more information about the assertion.

This policy is enforced only when attached with oauth2 client policies. Otherwise, it is ignored.

For more information about configuring the policy, see "oracle/oauth2_config_client_policy" on page 11-16.

B.1.1.8 oracle/http_saml20_token_bearer_client_policy

This policy includes a SAML Bearer V2.0 token in the HTTP header. The SAML token with confirmation method *Bearer* is created automatically. This policy can be enforced on any HTTP-based client endpoint.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_saml20_token_bearer_client_template. See "oracle/http_saml20_token_bearer_client_template" on page C-20 for more information about the assertion.

For more information about configuring the policy, see "oracle/http_saml20_token_bearer_client_policy" on page 11-17.

B.1.1.9 oracle/http_saml20_token_bearer_service_policy

This policy authenticates users using credentials provided in the SAML v2.0 token with confirmation method *Bearer* in the HTTP header. The credentials in the SAML token are authenticated against a SAML v2.0 login module. This policy can be enforced on any HTTP-based endpoint.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_saml20_token_bearer_ service_template. See "oracle/http_saml20_token_bearer_service_template" on page C-23 for more information about the assertion.

For more information about configuring the policy, see "oracle/http_saml20_bearer_token_service_policy" on page 11-19.

B.1.1.10 oracle/multi_token_rest_service_policy

This policy enforces one of the following authentication policies, based on the token sent by the client:

- HTTP Basic—Extracts username and password credentials from the HTTP header.
- SAML 2.0 *Bearer* token in the HTTP header—Extracts SAML 2.0 *Bearer* assertion in the HTTP header.
- HTTP OAM security—Verifies that the OAM agent has authenticated user and establishes identity.
- SPNEGO over HTTP security—Extracts Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) Kerberos token from the HTTP header.
- JWT token in the HTTP header—Extracts username from the JWT token in the HTTP header

This policy contains the following assertion templates as an OR group—meaning any one of the tokens can be sent by the client:

- oracle/wss_http_token_service_template. For more information, see "oracle/wss_http_token_client_template" on page C-27.
- oracle/http_saml20_token_bearer_service_template. For more information, see "oracle/http_saml20_token_bearer_service_template" on page C-23.
- oracle/http_oam_token_service_template. For more information, see "oracle/http_oam_token_service_template" on page C-10. (Provides OAM protection on the server-side only.)
- oracle/http_spnego_token_service_template. For more information, see "oracle/http_spnego_token_service_template" on page C-27.
- oracle/http_jwt_token_service_template. For more information, see "oracle/http_jwt_token_service_template" on page C-8.

B.1.1.11 oracle/wss_http_token_client_policy

The wss_http_token_client_policy includes credentials in the HTTP header for outbound client requests. This policy can be enforced on any HTTP-based client.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_client_template. See "oracle/wss_http_token_client_template" on page C-27 for more information about the assertion.

For more information about configuring the policy, see "oracle/wss_http_token_client_policy" on page 11-19.

B.1.1.12 oracle/wss_http_token_service_policy

The wss_http_token_service_policy uses the credentials in the HTTP header to authenticate users against the Oracle Platform Security Services identity store. This policy can be enforced on any HTTP-based endpoint.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_service_template. See "oracle/wss_http_token_service_template" on page C-29 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_http_token_service_policy" on page 11-21.

B.1.1.13 oracle/wss_username_token_client_policy

This policy includes credentials in the WS-Security UsernameToken SOAP header for all outbound SOAP request messages. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

Notes: Digest passwords are not supported in this release.

This policy is not secure; it transmits the password in clear text. You should use this policy in low security situations only, or when you know that the transport is protected using some other mechanism. Alternatively, consider using the SSL version of this policy, oracle/wss_username_token_over_ssl_client_policy.

This policy contains the following policy assertion: oracle/wss_username_token_client_template. See "oracle/wss_username_token_client_template" on page C-30 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_username_token_client_policy" on page 11-22.

B.1.1.14 oracle/wss_username_token_service_policy

This policy uses the credentials in the WS-Security UsernameToken SOAP header to authenticate users. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

Note: Digest passwords are not supported in this release.

This policy is not secure; it transmits the password in clear text. You should use this policy in low security situations only, or when you know that the transport is protected using some other mechanism. Alternatively, consider using the SSL version of this policy, oracle/wss_username_token_over_ssl_service_policy.

This policy contains the following policy assertion: oracle/wss_username_token_service_template. See "oracle/wss_username_token_service_template" on page C-33 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_username_token_service_policy" on page 11-23.

B.1.1.15 oracle/wss10_saml_token_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss10_saml_token_client_template. See "oracle/wss10_saml_token_client_template" on page C-34 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_client_policy" on page 11-23.

B.1.1.16 oracle/wss10_saml_token_service_policy

This policy authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. This policy can be enforced on any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss10_saml_token_service_template. See "oracle/wss10_saml_token_service_template" on page C-38 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_service_policy" on page 11-24.

B.1.1.17 oracle/wss10_saml20_token_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss10_saml20_token_client_template. See "oracle/wss10_saml20_token_client_template" on page C-39 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml20_token_client_policy" on page 11-25.

B.1.1.18 oracle/wss10_saml20_token_service_policy

This policy authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. This policy can be enforced on any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss10_saml20_token_service_template. See "oracle/wss10_saml20_token_service_template" on page C-43 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml20_token_service_policy" on page 11-26.

B.1.1.19 oracle/wss11_kerberos_token_client_policy

This policy includes a Kerberos token in the WS-Security header in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy is compatible with MIT and Active Directory KDCs. This policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_client_template. See "oracle/wss11_kerberos_token_with_message_protection_client_template" on page C-134 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_kerberos_token_client_policy" on page 11-26.

B.1.1.20 oracle/wss11_kerberos_token_service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy extracts the Kerberos token from the SOAP header and authenticates the user. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services. This policy is compatible with MIT and Active Directory KDCs. This policy can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-137 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_service_policy" on page 11-24.

B.1.1.21 oracle/wss_saml_token_bearer_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method *Bearer* is created automatically. This policy can be attached to any SOAP-based client.

This policy contains the following policy assertion: oracle/wss_saml_token_bearer_client_template. See "oracle/wss_saml_token_bearer_client_template" on page C-46 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_bearer_client_policy" on page 11-28.

B.1.1.22 oracle/wss_saml_or_username_token_service_policy

This policy enforces one of the following authentication policies, based on whether the client uses a SAML or username token, respectively:

- SAML token within WS-Security SOAP header using the sender-vouches confirmation type.
- WS-Security UsernameToken SOAP header to authenticate users against the configured identity store.

This policy contains the following assertions, as an OR group—meaning either type of policy can be enforced by a client:

- oracle/wss_saml_token_service_template. See "oracle/wss10_saml_token_service_template" on page C-38 for more information about the assertion.
- oracle/wss_username_token_service_template. See "oracle/wss_username_token_service_template" on page C-33 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_service_policy" on page 11-24 and "oracle/wss_username_token_service_policy" on page 11-23.

B.1.1.23 oracle/wss_saml_bearer_or_username_token_service_policy

This policy enforces one of the following authentication policies, based on whether the client uses a SAML or username token, respectively:

- SAML token within WS-Security SOAP header using the bearer confirmation type.

- WS-Security UsernameToken SOAP header to authenticate users against the configured identity store.

This policy contains the following assertions as an OR group—meaning either type of policy can be enforced by a client:

- oracle/wss_saml_token_bearer_template. See "oracle/wss_saml_token_bearer_service_template" on page C-73 for more information about the assertion.
- oracle/wss_username_token_template. See ""oracle/wss_username_token_service_template" on page C-33 for more information about the assertion.

B.1.2 Message Protection Only Policies

Table B–3 summarizes the policies that enforce message protection only, and indicates whether the policy is enforced at the transport layer or SOAP header.

Table B–3 Message-Protection Only Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss10_message_protection_client_policy	oracle/wss10_message_protection_service_policy	No	No	No	Yes
oracle/wss11_message_protection_client_policy	oracle/wss11_message_protection_service_policy	No	No	No	Yes

B.1.2.1 oracle/wss10_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses the WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_message_protection_client_template. See "oracle/wss10_message_protection_client_template" on page C-51 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_message_protection_client_policy" on page 11-28.

B.1.2.2 oracle/wss10_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

The messages are protected using WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_message_protection_service_template. See "oracle/wss10_message_protection_service_template" on page C-53 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_message_protection_service_policy" on page 11-30.

B.1.2.3 oracle/wss11_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_message_protection_client_template. See "oracle/wss11_message_protection_client_template" on page C-55 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_message_protection_client_policy" on page 11-31.

B.1.2.4 oracle/wss11_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_message_protection_service_template. See "oracle/wss11_message_protection_service_template" on page C-58 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_message_protection_service_policy" on page 11-33.

B.1.3 Message Protection and Authentication Policies

Table B–4 summarizes the policies that enforce both message protection and authentication but do not conform to the WS-Security 1.0 or 1.1 standard. The table indicates whether the policy is enforced at the transport layer or SOAP header.

Table B–4 Message Protection and Authentication Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss_http_token_over_ssl_client_policy	oracle/wss_http_token_over_ssl_service_policy	Yes	No	Yes	No
Attach one of the following: ▪ oracle/wss_saml_token_over_ssl_client_policy ▪ oracle/wss_username_token_over_ssl_client_policy	oracle/wss_saml_or_username_token_over_ssl_service_policy	No	Yes	Yes	No
oracle/wss_saml_token_bearer_over_ssl_client_policy	oracle/wss_saml_token_bearer_over_ssl_service_policy	No	Yes	Yes	No
oracle/wss_saml_token_over_ssl_client_policy	oracle/wss_saml_token_over_ssl_service_policy	No	Yes	Yes	No
oracle/wss_saml20_token_over_ssl_client_policy	oracle/wss_saml20_token_over_ssl_service_policy	No	Yes	Yes	No
oracle/wss_username_token_over_ssl_client_policy	oracle/wss_username_token_over_ssl_service_policy	No	Yes	Yes	No
oracle/wss10_saml_hok_with_message_protection_client_policy	oracle/wss10_saml_hok_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss10_saml_token_with_message_integrity_client_policy	oracle/wss10_saml_token_with_message_integrity_service_policy	No	Yes	No	Yes
oracle/wss10_saml_token_with_message_protection_client_policy	oracle/wss10_saml_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss10_saml20_token_with_message_protection_client_policy	oracle/wss10_saml20_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy	oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy	No	Yes	No	Yes
oracle/wss10_username_id_propagation_with_msg_protection_client_policy	oracle/wss10_username_id_propagation_with_msg_protection_service_policy	No	Yes	No	Yes

Table B–4 (Cont.) Message Protection and Authentication Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss10_username_token_with_message_protection_client_policy	oracle/wss10_username_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy	oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy	No	Yes	No	Yes
oracle/wss10_x509_token_with_message_protection_client_policy	oracle/wss10_x509_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss11_kerberos_token_with_message_protection_client_policy	oracle/wss11_kerberos_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy	oracle/wss11_kerberos_token_with_message_protection_basic128__service_policy	No	Yes	No	Yes
Attach one of the following: - oracle/wss11_saml_token_with_message_protection_client_policy - oracle/wss11_username_token_with_message_protection_client_policy - oracle/wss_saml_token_bearer_over_ssl_client_policy - oracle/wss_username_token_over_ssl_client_policy - oracle/wss_http_token_over_ssl_client_policy - oracle/http_jwt_token_over_ssl_client_policy	oracle/wss11_saml_or_username_token_with_message_protection_service_policy	No	Yes	No	Yes

Table B–4 (Cont.) Message Protection and Authentication Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss11_saml_token_with_message_protection_client_policy	oracle/wss11_saml_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss11_saml20_token_with_message_protection_client_policy	oracle/wss11_saml20_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss11_saml_token_with_identity_switch_message_protection_client_policy	oracle/wss11_saml_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss11_username_token_with_message_protection_client_policy	oracle/wss11_username_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss11_x509_token_with_message_protection_client_policy	oracle/wss11_x509_token_with_message_protection_service_policy	No	Yes	No	Yes

B.1.3.1 oracle/http_basic_auth_over_ssl_client_policy

This policy includes credentials in the HTTP header for outbound client requests. This policy verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be enforced on any HTTP-based client endpoint.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_over_ssl_service_template. See "oracle/wss_http_token_over_ssl_client_template" on page C-70 for more information about the assertion.

For information about configuring the policy, see "oracle/http_basic_auth_over_ssl_service_policy" on page 11-36.

B.1.3.2 oracle/http_basic_auth_over_ssl_service_policy

This policy uses the credentials in the HTTP header to authenticate users against the Oracle Platform Security Services identity store. This policy verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be enforced on any HTTP-based endpoint.

> **Note:** This policy functions similarly to oracle/wss_http_token_over_ssl_service_policy. The only difference is that `oracle/wss_http_token_over_ssl_service_policy` enables the `include-timestamp` attribute in the `require-tls` element to prevent replay attacks, a feature that is not applicable to RESTful services. For more information about the `require-tls` element, see "orasp:require-tls" on page D-47.
>
> Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_over_ssl_service_template. See "oracle/wss_http_token_over_ssl_service_template" on page C-72 for more information about the assertion.

For information about configuring the policy, see "oracle/http_basic_auth_over_ssl_service_policy" on page 11-36.

B.1.3.3 oracle/http_jwt_token_over_ssl_client_policy

This policy includes a JWT token in the HTTP header. The JWT token is created automatically. The issuer name and subject name are provided either programmatically or declaratively through the policy. You can specify the audience restriction condition for this policy.

This policy also verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused.

This policy can be enforced on any HTTP-based client endpoint.

This policy contains the following policy assertion: oracle/http_jwt_token_over_ssl_client_template. See "oracle/http_jwt_token_over_ssl_client_template" on page C-62 for more information about the assertion.

For information about configuring the policy, see "oracle/http_jwt_token_client_policy" on page 11-7.

B.1.3.4 oracle/http_jwt_token_over_ssl_service_policy

This policy authenticates users using the username provided in the JWT token in the HTTP header. This policy also verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused.

This policy can be applied to any HTTP-based endpoint.

This policy contains the following policy assertion: oracle/http_jwt_token_over_ssl_service_template. See "oracle/http_jwt_token_over_ssl_service_template" on page C-67 for more information about the assertion.

For information about configuring the policy, see "oracle/http_jwt_token_client_policy" on page 11-7.

B.1.3.5 oracle/http_oauth2_token_over_ssl_client_policy

This policy includes the OAuth2 access token in the HTTP header. The access token (AT) is obtained from the Mobile & Social OAuth2 Server. You can attach this policy to any HTTP-based client.

The policy verifies that the outbound transport protocol is HTTPS. If a non-HTTPS transport protocol is used, the request is refused.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_oauth2_token_over_ssl_client_template. See "oracle/http_oauth2_token_over_ssl_client_template" on page C-69 for more information about the assertion.

This policy must always be attached with oracle/oauth2_config_client_policy.

For more information about configuring the policy, see "oracle/http_oauth2_token_over_ssl_client_policy" on page 11-40.

B.1.3.6 oracle/http_oauth2_token_opc_oauth2_over_ssl_client_policy

This policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the Mobile & Social OAuth2 Server. The property `oracle.oauth2.service` is set to true by default, which ensures that the client ID is used as the issuer for the user and client JWT tokens for the OAuth2 server. If `scope` has no value, (the default), the protocol, host and port (if available) are obtained from the service URL and used.

The policy verifies that the outbound transport protocol is HTTPS. If a non-HTTPS transport protocol is used, the request is refused. You can attach this policy to any HTTP-based client.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_oauth2_token_over_ssl_client_template. See "oracle/http_oauth2_token_over_ssl_client_template" on page C-69 for more information about the assertion.

This policy must always be attached with oracle/oauth2_config_client_policy.

For more information about configuring the policy, see "oracle/http_oauth2_token_opc_oauth2_over_ssl_client_policy" on page 11-46.

B.1.3.7 oracle/http_oauth2_token_identity_switch_over_ssl_client_policy

This policy is similar to the policy oracle/http_oauth2_token_over_ssl_client_policy, with the subject.precedence property set to false by default.

This policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the Mobile and Social OAuth2 Server.) It also verifies that the outbound transport protocol is HTTPS. If a non-HTTPS transport protocol is used, the request is refused.

This policy performs dynamic identity switching by propagating a different identity than the one based on the authenticated subject. This policy can be attached to any HTTP-based SOAP or REST client.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_oauth2_token_over_ssl_client_template. See "oracle/http_oauth2_token_over_ssl_client_template" on page C-69 for more information about the assertion.

This policy must always be attached with oracle/oauth2_config_client_policy.

For more information about configuring the policy, see "oracle/http_oauth2_token_identity_switch_over_ssl_client_policy" on page 11-43.

B.1.3.8 oracle/http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_policy

This policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the OAuth Server. It also verifies that the outbound transport protocol is

HTTPS. If a non-HTTPS transport protocol is used, the request is refused. This policy can be attached to any HTTP-based SOAP or REST client, invoking the service over SSL.

This policy also performs dynamic identity switching by propagating a different identity than the one based on the authenticated subject.

The `subject.precedence` property set to false by default. The `oracle.oauth2.service` property is set to true by default, which ensures that the client ID is used as the issuer for the user and client JWT tokens for the OAuth2 server.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_oauth2_token_over_ssl_client_template. See "oracle/http_oauth2_token_over_ssl_client_template" on page C-69 for more information about the assertion.

This policy must always be attached with oracle/oauth2_config_client_policy.

For more information about configuring the policy, see "oracle/http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_policy" on page 11-50.

B.1.3.9 oracle/http_saml20_token_bearer_over_ssl_client_policy

This policy includes a SAML Bearer v2.0 token in the HTTP header. The SAML token with confirmation method *Bearer* is created automatically. The policy verifies that the transport protocol provides SSL message protection. This policy can be attached to any HTTP-based client endpoint.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_saml20_token_bearer_client_template. See "oracle/http_saml20_token_bearer_client_template" on page C-20 for more information about the assertion.

For more information about configuring the policy, see "oracle/http_saml20_bearer_token_over_ssl_client_policy" on page 11-53.

B.1.3.10 oracle/http_saml20_token_bearer_over_ssl_service_policy

This policy authenticates users using credentials provided in the SAML v2.0 token with confirmation method *Bearer* in the HTTP header. The credentials in the SAML token are authenticated against a SAML v2.0 login module. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any HTTP-based endpoint.

This policy contains the following assertion template, which defines the settings and configuration properties for the policy assertion: oracle/http_saml20_token_bearer_service_template. See "oracle/http_saml20_token_bearer_service_template" on page C-23 for more information about the assertion.

For more information about configuring the policy, see "oracle/http_saml20_bearer_token_over_ssl_service_policy" on page 11-54.

B.1.3.11 oracle/multi_token_over_ssl_rest_service_policy

This policy enforces one of the following authentication policies, based on the token sent by the client:

- HTTP Basic over SSL—Extracts username and password credentials from the HTTP header.
- SAML 2.0 *Bearer* token in the HTTP header over SSL—Extracts SAML 2.0 *Bearer* assertion in the HTTP header.

- HTTP OAM security (non-SSL)—Verifies that the OAM agent has authenticated user and establishes identity. (Provides non-SSL OAM protection on the server-side only.)
- SPNEGO over HTTP security (non-SSL)—Extracts SPNEGO Kerberos token information from the HTTP header. (Provides non-SSL protection only.)
- JWT token in the HTTP header over SSL—Extracts username from the JWT token in the HTTP header

This policy contains the following assertion templates as an OR group—meaning any one of the tokens can be sent by the client:

- oracle/wss_http_token_over_ssl_service_template. For more information, see "oracle/wss_http_token_over_ssl_service_template" on page C-72.
- oracle/http_saml20_token_over_ssl_bearer_service_policy. For more information about configuring this policy, see "oracle/http_saml20_token_bearer_service_template" on page C-23.
- oracle/http_oam_token_service_template. (Provides non-SSL OAM protection on the server-side only.) For more information, see "oracle/http_oam_token_service_template" on page C-10.
- oracle/http_spnego_token_service_template. (Provides non-SSL protection only.) For more information, see "oracle/http_spnego_token_service_template" on page C-27.
- oracle/http_jwt_token_over_ssl_service_template. For more information, see "oracle/http_jwt_token_over_ssl_service_template" on page C-67.

B.1.3.12 oracle/wss_http_token_over_ssl_client_policy

This policy includes credentials in the HTTP header for outbound client requests and authenticates users against the Oracle Platform Security Services identity store. This policy also verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be enforced on any HTTP-based client.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_over_ssl_client_template. See "oracle/wss_http_token_over_ssl_client_template" on page C-70 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_http_token_over_ssl_client_policy" on page 11-55.

B.1.3.13 oracle/wss_http_token_over_ssl_service_policy

This policy extracts the credentials in the HTTP header and authenticates users against the Oracle Platform Security Services identity store. This policy verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be enforced on any HTTP-based endpoint.

Notes: This policy functions similarly to oracle/http_basic_auth_over_ssl_service_policy. The only difference is that `oracle/wss_http_token_over_ssl_service_policy` enables the `include-timestamp` attribute in the `require-tls` element to prevent replay attacks, which is not applicable to RESTful services. For more information about the `require-tls` element, see "orasp:require-tls" on page D-47.

Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_over_ssl_service_template. See "oracle/wss_http_token_over_ssl_service_template" on page C-72 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_http_token_over_ssl_service_policy" on page 11-56.

B.1.3.14 oracle/wss_saml_or_username_token_over_ssl_service_policy

This policy enforces message protection (integrity and confidentiality) and one of the following authentication policies, based on whether the client uses a SAML or username token, respectively:

- SAML token within WS-Security SOAP header using the sender-vouches confirmation type.
- WS-Security UsernameToken SOAP header to authenticate users against the configured identity store.

This policy contains the following assertions, as an OR group—meaning either type of policy can be enforced by a client:

- oracle/wss_saml_token_over_ssl_service_template. See "oracle/wss_saml_token_over_ssl_service_template" on page C-90 for more information about the assertion.
- oracle/wss_username_token_over_ssl_service_template. See "oracle/wss_username_token_over_ssl_service_template" on page C-99 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_over_ssl_service_policy" on page 11-61 and "oracle/wss_username_token_over_ssl_service_policy" on page 11-64.

B.1.3.15 oracle/wss_saml_token_bearer_identity_switch_client_policy

Performs dynamic identity switching by propagating a different identity than the one based on the authenticated subject. This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method Bearer is created automatically. This policy can be attached to any SOAP-based client.

This policy contains the following assertion: oracle/wss_saml_token_bearer_client_template. See "oracle/wss_saml_token_bearer_client_template" on page C-46 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_bearer_identity_switch_client_policy" on page 11-57.

B.1.3.16 oracle/wss_saml_token_bearer_over_ssl_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method *Bearer* is created automatically. The policy also verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based client.

This policy contains the following policy assertion: oracle/wss_saml_token_bearer_over_ssl_client_template. See "oracle/wss_saml_token_bearer_over_ssl_client_template" on page C-75 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_bearer_over_ssl_client_policy" on page 11-57.

B.1.3.17 oracle/wss_saml_token_bearer_over_ssl_service_policy

This policy authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss_saml_token_bearer_over_ssl_service_template. See "oracle/wss_saml_token_bearer_over_ssl_service_template" on page C-79 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_bearer_over_ssl_service_policy" on page 11-58.

B.1.3.18 oracle/wss_saml20_token_bearer_over_ssl_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method *Bearer* is created automatically. The policy also verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based client.

This policy contains the following policy assertion: oracle/wss_saml20_token_bearer_over_ssl_client_template. See "oracle/wss_saml20_token_bearer_over_ssl_client_template" on page C-81 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml20_token_bearer_over_ssl_client_policy" on page 11-59.

B.1.3.19 oracle/wss_saml20_token_bearer_over_ssl_service_policy

This policy authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss_saml20_token_bearer_over_ssl_service_template. See "oracle/wss_saml20_token_bearer_over_ssl_service_template" on page C-85 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml20_token_bearer_over_ssl_service_policy" on page 11-60.

B.1.3.20 oracle/wss_saml_token_over_ssl_client_policy

This policy includes SAML tokens in outbound WS-Security SOAP headers using the sender-vouches confirmation type. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss_saml_token_over_ssl_client_template. See "oracle/wss_saml_token_over_ssl_client_template" on page C-86 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_over_ssl_client_policy" on page 11-60.

B.1.3.21 oracle/wss_saml_token_over_ssl_service_policy

This policy enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type. The SAML token is mapped to a user in the configured identity store. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss_saml_token_over_ssl_service_template. See "oracle/wss_saml_token_over_ssl_service_template" on page C-90 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_over_ssl_service_policy" on page 11-61.

B.1.3.22 oracle/wss_saml20_token_over_ssl_client_policy

This policy includes SAML tokens in outbound WS-Security SOAP headers using the sender-vouches confirmation type. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss_saml20_token_over_ssl_client_template. See "oracle/wss_saml20_token_over_ssl_client_template" on page C-91 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml20_token_over_ssl_client_policy" on page 11-62.

B.1.3.23 oracle/wss_saml20_token_over_ssl_service_policy

This policy enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type. The SAML token is mapped to a user in the configured identity store. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss_saml20_token_over_ssl_service_template. See "oracle/wss_saml20_token_over_ssl_service_template" on page C-95 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml20_token_over_ssl_service_policy" on page 11-63.

B.1.3.24 oracle/wss_username_token_over_ssl_client_policy

This policy includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages. The policy verifies that the transport protocol provides SSL message protection. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

Note: Digest passwords are not supported in this release.

This policy contains the following policy assertion: oracle/wss_username_token_over_ssl_client_template. See "oracle/wss_username_token_over_ssl_client_template" on page C-96 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_username_token_over_ssl_client_policy" on page 11-64.

B.1.3.25 oracle/wss_username_token_over_ssl_service_policy

This policy uses the credentials in the WS-Security UsernameToken SOAP header to authenticate users against the Oracle Platform Security Services configured identity store. The policy verifies that the transport protocol provides SSL message protection. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

> **Note:** Digest passwords are not supported in this release.

This policy contains the following policy assertion: oracle/wss_username_token_over_ssl_service_template. See "oracle/wss_username_token_over_ssl_service_template" on page C-99 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_username_token_over_ssl_service_policy" on page 11-64.

B.1.3.26 oracle/wss10_saml_hok_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and SAML holder of key based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with holder of key confirmation.

The policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_saml_hok_with_message_protection_client_template. See "oracle/wss10_saml_hok_token_with_message_protection_client_template" on page C-100 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_hok_token_with_message_protection_client_policy" on page 11-65.

B.1.3.27 oracle/wss10_saml_hok_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML holder of key based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_saml_hok_with_message_protection_service_template. See "oracle/wss10_saml_hok_token_with_ message_protection_service_template" on page C-105 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_hok_token_with_message_protection_service_policy" on page 11-66.

B.1.3.28 oracle/wss10_saml_token_with_message_integrity_client_policy

This policy provides message-level integrity and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies and SHA-1 hashing algorithm for message integrity. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-107 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_integrity_client_policy" on page 11-67.

B.1.3.29 oracle/wss10_saml_token_with_message_integrity_service_policy

This policy enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. It extracts the SAML token from the WS-Security binary security token or the current Java Authentication and Authorization Service (JAAS) subject, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies and SHA-1 hashing algorithm for message integrity. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-113 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_integrity_service_policy" on page 11-68.

B.1.3.30 oracle/wss10_saml_token_with_message_protection_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-107 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_protection_client_policy" on page 11-69.

B.1.3.31 oracle/wss10_saml_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-113 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_protection_service_policy" on page 11-70.

B.1.3.32 oracle/wss10_saml20_token_with_message_protection_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_saml20_token_with_message_protection_client_template. See "oracle/wss10_saml20_token_with_message_protection_client_template" on page C-115 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml20_token_with_message_protection_client_policy" on page 11-71.

B.1.3.33 oracle/wss10_saml20_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_saml20_token_with_message_protection_service_template. See "oracle/wss10_saml20_token_with_message_protection_service_template" on page C-121 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml20_token_with_message_protection_service_policy" on page 11-72.

B.1.3.34 oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

The policy uses WS-Security's Basic 256 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-256 bit encryption. This policy uses Subject Key Identifier (ski) reference mechanism for encryption key in the request and for both signature and encryption keys in the response. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-107 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_protection_client_policy" on page 11-69.

B.1.3.35 oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

The policy uses WS-Security's Basic 256 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-256 bit encryption. This policy uses Subject Key Identifier (ski) reference mechanism for encryption key in the request and for both signature and encryption keys in the response. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-113 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_protection_service_policy" on page 11-70.

B.1.3.36 oracle/wss10_username_id_propagation_with_msg_protection_client_policy

This policy provides message protection (integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Credentials (only username) are included in outbound SOAP request messages via a WS-Security UsernameToken header. No password is included.This policy can be enforced on any SOAP-based client.

Message protection is provided using WS-Security's Basic128 suite of asymmetric key technologies. Specifically RSA key mechanisms for confidentiality, SHA-1 hashing algorithm for integrity and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-123 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_id_propagation_with_msg_protection_client_policy" on page 11-77.

B.1.3.37 oracle/wss10_username_id_propagation_with_msg_protection_service_policy

This policy enforces message level protection (i.e., integrity and confidentiality) and identity propagation for inbound SOAP requests using mechanisms described in WS-Security 1.0. This policy can be enforced on any SOAP-based endpoint.

Message protection is provided using WS-Security 1.0's Basic128 suite of asymmetric key technologies. Specifically RSA key mechanisms for confidentiality, SHA-1 hashing algorithm for integrity and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_username_id_propagation_with_msg_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-127 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_id_propagation_with_msg_protection_service_policy" on page 11-78.

B.1.3.38 oracle/wss10_username_token_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-123 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_token_with_message_protection_client_policy" on page 11-79.

B.1.3.39 oracle/wss10_username_token_with_message_protection_service_policy

This policy enforces message protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-127 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_token_with_message_protection_service_policy" on page 11-80.

B.1.3.40 oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy

This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

This policy uses WS-Security's Basic 256 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-256 bit encryption. This policy uses Subject Key Identifier (ski) reference mechanism for encryption key in the request and for both signature and encryption keys in the response. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-123 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_token_with_message_protection_client_policy" on page 11-79.

B.1.3.41 oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy

This policy enforces message protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

This policy uses WS-Security's Basic 256 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-256 bit encryption. This policy uses Subject Key Identifier (ski) reference mechanism for encryption key in the request and for both signature and encryption keys in the response. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-127 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_token_with_message_protection_service_policy" on page 11-80.

B.1.3.42 oracle/wss10_x509_token_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and certificate credential population for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_x509_token_with_message_protection_client_template. See "oracle/wss10_x509_token_with_message_protection_client_template" on page C-129 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_x509_token_with_message_protection_client_policy" on page 11-84.

B.1.3.43 oracle/wss10_x509_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss10_x509_token_with_message_protection_service_template. See "oracle/wss10_x509_token_with_message_protection_service_template" on page C-132 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_x509_token_with_message_protection_service_policy" on page 11-85.

B.1.3.44 oracle/wss11_kerberos_token_with_message_protection_client_policy

This policy includes a Kerberos token in the WS-Security header, and uses Kerberos keys to guarantee message integrity and confidentiality, in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy is compatible with MIT and Active Directory KDCs. This policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_with_message_protection_client_template. See "oracle/wss11_kerberos_token_with_ message_protection_client_template" on page C-134 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_kerberos_token_with_message_protection_client_policy" on page 11-85.

B.1.3.45 oracle/wss11_kerberos_token_with_message_protection_service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy is compatible with MIT and Active Directory KDCs. This policy can be attached to any SOAP-based endpoint.

This policy extracts the Kerberos token from the SOAP header and authenticates the user, and it enforces message integrity and confidentiality using Kerberos keys. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_with_message_protection_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-137 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_kerberos_token_with_message_protection_service_policy" on page 11-86.

B.1.3.46 oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy

This policy includes a Kerberos token in the WS-Security header, and uses Kerberos keys to guarantee message integrity and confidentiality, in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy is compatible with Active Directory KDCs. This policy can be enforced on any SOAP-based client.

This policy uses the WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_with_message_protection_client_template. See "oracle/wss11_kerberos_token_with_message_protection_client_template" on page C-134 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy" on page 11-87.

B.1.3.47 oracle/wss11_kerberos_token_with_message_protection_basic128__service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy is compatible with Active Directory KDCs. This policy can be attached to any SOAP-based endpoint.

This policy uses the WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy extracts the Kerberos token from the SOAP header and authenticates the user, and it enforces message integrity and confidentiality using Kerberos keys. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_with_message_protection_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-137 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy" on page 11-88.

B.1.3.48 oracle/wss11_saml_token_with_message_protection_client_policy

This policy enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1. A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_saml_token_with_message_protection_client_template. See "oracle/wss11_saml_token_with_message_protection_client_template" on page C-138 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_saml_token_with_message_protection_client_policy" on page 11-89.

B.1.3.49 oracle/wss11_saml20_token_with_message_protection_client_policy

This policy enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1. A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_saml20_token_with_message_protection_client_template. See "oracle/wss11_saml20_token_with_message_protection_client_template" on page C-145 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_saml20_token_with_message_protection_client_policy" on page 11-92.

B.1.3.50 oracle/wss11_saml_token_with_identity_switch_message_protection_client_policy

This policy performs dynamic identity switching by propagating a different identity than the one based on the authenticated subject. This policy can be attached to any SOAP-based client.

This policy enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security

1.1. A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_saml_token_with_message_protection_client_template. See "oracle/wss11_saml_token_with_message_protection_client_template" on page C-138 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy" on page 11-90.

B.1.3.51 oracle/wss11_saml_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_saml_token_with_message_protection_service_template. See "oracle/wss11_saml_token_with_message_protection_service_template" on page C-143 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_saml_token_with_message_protection_service_policy" on page 11-91.

B.1.3.52 oracle/wss11_saml20_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_saml20_token_with_message_protection_service_template. See "oracle/wss11_saml20_token_with_message_protection_service_template" on page C-151 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_saml20_token_with_message_protection_service_policy" on page 11-94.

B.1.3.53 oracle/wss11_saml_or_username_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and one of the following authentication policies, based on whether the client uses a SAML, username, or HTTP token, respectively:

- SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.
- Username token authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.
- SAML-based authentication using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header. Verifies that the transport protocol provides SSL message protection.
- Username token authentication using the credentials in the UsernameToken WS-Security SOAP header to authenticate users against the configured identity store. Verifies that the transport protocol provides SSL message protection.
- HTTP authentication using credentials extracted from the HTTP header to authenticate users against the configured identity store. Verifies that the transport protocol is HTTPS.
- JWT token authentication using the username extracted from the JWT token in the HTTP header. Verifies that the transport protocol is HTTPS.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following assertions, as an OR group—meaning either type of policy can be enforced by a client:

- oracle/wss11_saml_token_with_message_protection_service_template. For more information about the assertion, see "oracle/wss11_saml_token_with_message_protection_service_template" on page C-143.
- oracle/wss11_username_token_with_message_protection_service_template. For more information about the assertion, see "oracle/wss11_username_token_with_message_protection_service_template" on page C-157.
- oracle/wss_saml_token_bearer_over_ssl_service_template. For more information about the assertion, see "oracle/wss_saml_token_bearer_over_ssl_service_template" on page C-79.
- oracle/wss_username_token_over_ssl_service_template. For more information about the assertion, see "oracle/wss_username_token_over_ssl_service_template" on page C-99.
- oracle/wss_http_token_over_ssl_service_template. For more information about the assertion, see "oracle/wss_http_token_over_ssl_service_template" on page C-72.
- oracle/http_jwt_token_over_ssl_service_template. For more information, see "oracle/http_jwt_token_over_ssl_service_template" on page C-67.

For information about configuring the policy, see the following:

- "oracle/wss11_saml_token_with_message_protection_service_policy" on page 11-91

- "oracle/wss11_username_token_with_message_protection_service_policy" on page 11-96
- "oracle/wss_saml_token_bearer_over_ssl_service_policy" on page 11-58
- "oracle/wss_username_token_over_ssl_service_policy" on page 11-64
- "oracle/wss_http_token_over_ssl_service_policy" on page 11-56.
- "oracle/http_jwt_token_over_ssl_service_policy" on page 11-39

B.1.3.54 oracle/wss11_username_token_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

> **Note:** Digest passwords are not supported in this release.

The Web service consumer inserts username and password credentials, and signs and encrypts the outgoing SOAP message. The Web service provider decrypts and verifies the message and the signature.

To prevent replay attacks, the assertion provides the option to include time stamps and verification by the Web service provider. The message can be protected with ciphers of different strengths.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_username_token_with_message_protection_client_template. See "oracle/wss11_username_token_with_message_protection_client_template" on page C-153 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_username_token_with_message_protection_client_policy" on page 11-95.

B.1.3.55 oracle/wss11_username_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. Both plain text and digest mechanisms are supported.

> **Note:** Digest passwords are not supported in this release.

The Web service consumer inserts username and password credentials, and signs and encrypts the outgoing SOAP message. The Web service provider decrypts and verifies the message and the signature. This policy can be attached to any SOAP-based endpoint.

To prevent replay attacks, the assertion provides the option to include time stamps and verification by the Web service provider. The message can be protected with ciphers of different strengths.

Note: Digest passwords are not supported in this release.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_username_token_with_message_protection_service_template. See "oracle/wss11_username_token_with_message_protection_service_template" on page C-157 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_username_token_with_message_protection_service_policy" on page 11-96.

B.1.3.56 oracle/wss11_x509_token_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_x509_token_with_message_protection_client_template. See "oracle/wss11_x509_token_with_message_protection_client_template" on page C-159 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_x509_token_with_message_protection_client_policy" on page 11-96.

B.1.3.57 oracle/wss11_x509_token_with_message_protection_service_policy

This policy enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-191.

This policy contains the following policy assertion: oracle/wss11_x509_token_with_message_protection_service_template. See "oracle/wss11_x509_token_with_message_protection_service_template" on page C-162 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_x509_token_with_message_protection_service_policy" on page 11-97.

B.1.4 WS-Trust Policies

Table B–5 summarizes the WS-Trust policies.

Table B–5 WS-Trust Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/sts_trust_config_client_policy	oracle/sts_trust_config_service_policy	No	No	No	No
oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy	oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy	Yes	No	Yes	No
oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy	oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss11_sts_issued_saml_with_message_protection_client_policy		No	Yes	No	Yes

B.1.4.1 oracle/sts_trust_config_service_policy

This policy provides STS configuration information that is used to invoke an STS for token exchange.

This policy contains the following policy assertion: oracle/sts_trust_config_template. See "oracle/sts_trust_config_client_template" on page C-165 for more information about the assertion.

For information about configuring the policy, see "oracle/sts_trust_config_service_policy" on page 11-110.

B.1.4.2 oracle/sts_trust_config_client_policy

This policy provides STS configuration information that is used to invoke an STS for token exchange.

This policy contains the following policy assertion: oracle/sts_trust_config_template. See "oracle/sts_trust_config_client_template" on page C-165 for more information about the assertion.

For information about configuring the policy, see "oracle/sts_trust_config_client_policy" on page 11-111.

B.1.4.3 oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy

This policy inserts the SAML Bearer assertion issued by a trusted STS (Security Token Service). Messages are protected using SSL.

This policy contains the following policy assertion: oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template. See "oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template" on page C-167 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy" on page 11-113.

B.1.4.4 oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy

This policy authenticates a SAML Bearer assertion issued by a trusted STS (Security Token Service). Messages are protected using SSL.

This policy contains the following policy assertion: oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template. See "oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template" on page C-171 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy" on page 11-115.

B.1.4.5 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy

This policy inserts a SAML HOK assertion issued by a trusted STS (Security Token Service). Messages are protected using proof key material provided by STS.

This policy contains the following policy assertion: oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template. See "oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template" on page C-172 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy" on page 11-116.

B.1.4.6 oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy

This policy inserts a SAML HOK assertion issued by a trusted STS (Security Token Service). Messages are protected using proof key material provided by STS.

This policy contains the following policy assertion: oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template. See "oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template" on page C-177 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy" on page 11-119.

B.1.4.7 oracle/wss11_sts_issued_saml_with_message_protection_client_policy

This policy inserts a SAML sender vouches assertion issued by a trusted STS (Security Token Service). Messages are protected using the client's private key.

This policy contains the following policy assertion: oracle/wss11_sts_issued_saml_with_message_protection_client_template. See "oracle/wss11_sts_issued_saml_with_message_protection_client_template" on page C-179 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_sts_issued_saml_with_message_protection_client_policy" on page 11-120.

B.1.5 Authorization Only Policies

Table B–6 summarizes the security policies that enforce authorization, and indicates whether the policy is enforced at the transport layer or SOAP header.

Note: The authorization polices can follow any authentication policy where the Subject is established.

You cannot attach both a permitall and denyall policy to the same Web service.

Table B–6 Authorization Only Policies

Client Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/binding_authorization_denyall_policy	No	Yes	No	No
oracle/binding_authorization_permitall_policy	No	Yes	No	No
oracle/binding_permission_authorization_policy	No	Yes	No	No
oracle/component_authorization_denyall_policy	No	Yes	No	No
oracle/component_authorization_permitall_policy	No	Yes	No	No
oracle/component_permission_authorization_policy	No	Yes	No	No
oracle/whitelist_authorization_policy	No	Yes	No	No

B.1.5.1 oracle/binding_authorization_denyall_policy

This policy provides simple role-based authorization for the request based on the authenticated Subject at the SOAP binding level. This policy denies all users with any roles. It should follow an authentication policy where the Subject is established and can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/binding_authorization_template. See "oracle/binding_authorization_template" on page C-183 for more information about the assertion.

For information about configuring the policy, see "oracle/binding_authorization_denyall_policy" on page 11-102.

B.1.5.2 oracle/binding_authorization_permitall_policy

This policy provides a simple role-based authorization for the request based on the authenticated Subject at the SOAP binding level. This policy permits all users with any roles. It should follow an authentication policy where the Subject is established and can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/binding_authorization_template. See "oracle/binding_authorization_template" on page C-183 for more information about the assertion.

For information about configuring the policy, see "oracle/binding_authorization_permitall_policy" on page 11-103.

B.1.5.3 oracle/binding_permission_authorization_policy

This policy provides simple permission-based authorization for the request based on the authenticated Subject at the SOAP binding level. This policy ensures that the Subject has permission to perform the operation. This policy should follow an authentication policy where the Subject is established and can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/binding_permission_authorization_template. See "oracle/component_permission_authorization_template" on page C-187 for more information about the assertion.

For information about configuring the policy, see "oracle/binding_permission_authorization_policy" on page 11-104.

B.1.5.4 oracle/component_authorization_denyall_policy

This policy provides simple role-based authorization for the request based on the authenticated Subject at the SOAP binding level. This policy denies all users with any roles. It should follow an authentication policy where the Subject is established and can be attached to any SCA-based endpoint.

This policy contains the following policy assertion: oracle/component_authorization_template. See "oracle/component_authorization_template" on page C-186 for more information about the assertion.

For information about configuring the policy, see "oracle/component_authorization_denyall_policy" on page 11-104.

B.1.5.5 oracle/component_authorization_permitall_policy

This policy provides a simple role-based authorization policy based on the authenticated Subject. This policy permits all users with any roles. It should follow an authentication policy where the Subject is established and can be attached to any SCA-based endpoint.

This policy contains the following policy assertion: oracle/component_authorization_template. See "oracle/component_authorization_template" on page C-186 for more information about the assertion.

For information about configuring the policy, see "oracle/binding_authorization_permitall_policy" on page 11-103.

B.1.5.6 oracle/component_permission_authorization_policy

This policy provides a permission-based authorization policy based on the authenticated Subject. This policy ensures that the Subject has permission to perform the operation. This policy should follow an authentication policy where the Subject is established and can be attached to any SCA-based endpoint.

This policy contains the following policy assertion: oracle/component_permission_authorization_template. See "oracle/component_permission_authorization_template" on page C-187 for more information about the assertion.

For information about configuring the policy, see "oracle/component_permission_authorization_policy" on page 11-106.

B.1.5.7 oracle/whitelist_authorization_policy

This policy is a special case of role based authorization policy. This policy will let requests in only if authenticated token is SAML Sender-Vouches or if the user is in a particular role 'trustedEnterpriseRole' that established the user as a trusted entity or if the request is coming from within the private network. This policy can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/binding_authorization_template. See "oracle/binding_authorization_template" on page C-183 for more information about the assertion.

For information about configuring this policy, see "oracle/whitelist_authorization_policy" on page 11-107.

B.1.6 Authorization Policies—Oracle Entitlements Server

Table B–7 summarize the predefined Oracle WSM Oracle Entitlements Server (OES) security policies.

Table B–7 Oracle Entitlements Server Policies

Policy Name	Description
oracle/binding_oes_authorization_policy	Sets user authorization based on the policy defined in Oracle Entitlements Server.
oracle/binding_oes_masking_policy	Does response masking based on the policy defined in Oracle Entitlements Server.
oracle/component_oes_authorization_policy	Sets user authorization based on the policy defined in Oracle Entitlements Server.

B.1.6.1 oracle/binding_oes_authorization_policy

This policy sets authorization based on the policy defined in Oracle Entitlements Server (OES). Authorization is based on attributes, the current authenticated subject, and the web service action invoked by the client. This policy is used for fine-grained authorization on any operation on the web service.

This policy should follow an authentication policy where the subject is established. You can attach this policy to any SOAP-based or REST-based endpoint.

This policy contains the following policy assertion: oracle/binding_oes_authorization_template. See "oracle/binding_oes_authorization_template" on page C-188 for more information about the assertion.

For information about configuring the policy, see "Configuring Fine-Grained Authorization Using Oracle Entitlements Server" on page 11-148.

B.1.6.2 oracle/binding_oes_masking_policy

This policy does response masking based on the policy defined in OES. Masking is based on attributes, the current authenticated subject, and the web service action invoked by the client. This template is used for fine-grained masking on any operation of a web service.

This policy should follow an authentication policy where the subject is established. You can attach this policy to any SOAP endpoint.

This policy contains the following policy assertion: oracle/binding_oes_masking_template. See "oracle/binding_oes_masking_template" on page C-190 for more information about the assertion.

For information about configuring the policy, see "Configuring Fine-Grained Authorization Using Oracle Entitlements Server" on page 11-148.

B.1.6.3 oracle/component_oes_authorization_policy

This policy does user authorization based on the policy defined in Oracle Entitlements Server (OES).

This policy contains the following policy assertion: oracle/component_oes_authorization_template. See "oracle/component_oes_authorization_template" on page C-190 for more information about the assertion.

For information about configuring the policy, see "Configuring Fine-Grained Authorization Using Oracle Entitlements Server" on page 11-148.

B.2 WS-Addressing Policies

This section describes the predefined WS-Addressing policies.

Note: WS-Addressing policies are not supported for WebLogic Web services.

B.2.1 oracle/wsaddr_policy

This policy causes the platform to check inbound messages for the presence of WS-Addressing headers conforming to the W3C 2005 Final WS-Addressing Policy standard. In addition, it causes the platform to include a WS-Addressing header in outbound SOAP messages. For information about configuring the policy, see "oracle/wsaddr_policy" on page 11-109.

B.3 MTOM Attachment Policies

This section describes the predefined MTOM policies.

Note: MTOM policies are not supported for WebLogic Web services.

B.3.1 oracle/wsmtom_policy

This Message Transmission Optimization Mechanism (MTOM) policy rejects inbound messages that are not in MTOM format and verifies that outbound messages are in MTOM format. MTOM refers to specifications `http://www.w3.org/TR/2005/REC-soap12-mtom-20050125` and `http://www.w3.org/Submission/2006/SUBM-soap11mtom10-20060405` for SOAP 1.2 and SOAP 1.1 bindings, respectively. For information about configuring the policy, see "oracle/wsmtom_policy" on page 11-123.

B.4 Reliable Messaging Policies

This section describes the predefined Reliable Messaging policies.

Note: Reliable messaging policies are not supported for WebLogic Web services.

B.4.1 oracle/wsrm10_policy

This policy provides support for version 1.0 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint. Full support for this feature may require additional programming. For information about configuring the policy, see "oracle/wsrm10_policy" on page 11-125.

B.4.2 oracle/wsrm11_policy

This policy provides support for version 1.1 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint. Full support for this feature may require additional programming. For information about configuring the policy, see "oracle/wsrm11_policy" on page 11-126.

B.5 Management Policies

This section describes the predefined Management policies.

Note: Management policies are not supported for WebLogic Web services.

B.5.1 oracle/log_policy

This policy causes the request, response, and fault messages to be sent to a message log. For information about configuring the policy, see "oracle/log_policy" on page 11-127.

This policy contains the following policy assertion: oracle/security_log_template. See "oracle/security_log_template" on page C-195 for more information about the assertion.

B.6 No Behavior Policies

This section describes the predefined no behavior policies. These policies provide the ability to effectively disable a policy attached globally in a policy set. Details for using these policies are provided in "Disabling a Globally Attached Policy" on page 9-30. There are no configuration properties available for these policies.

All of these policies use the same no behavior assertion.

Note: The no behavior policies are not supported for WebLogic Web services.

B.6.1 oracle/no_authentication_service_policy

This policy, when directly attached to a service endpoint or globally attached at a lower scope, effectively disables a globally attached authentication policy at a higher scope. If the globally attached policy contains any other assertions, in addition to the authentication assertion, those assertions are disabled also.

B.6.2 oracle/no_authentication_client_policy

This policy, when directly attached to a client endpoint or globally attached at a lower scope, effectively disables a globally attached authentication policy at a higher scope. If the globally attached policy contains any other assertions, in addition to the authentication assertion, those assertions are disabled also.

B.6.3 oracle/no_messageprotection_service_policy

This policy, when directly attached to a service endpoint or globally attached at a lower scope, effectively disables a globally attached message protection policy at a higher scope. If the globally attached policy contains any other assertions, in addition to the message protection assertion, those assertions are disabled also.

B.6.4 oracle/no_messageprotection_client_policy

This policy, when directly attached to a client endpoint or globally attached at a lower scope, effectively disables a globally attached message protection policy at a higher scope. If the globally attached policy contains any other assertions, in addition to the message protection assertion, those assertions are disabled also.

B.6.5 oracle/no_authorization_service_policy

This policy, when directly attached to a service endpoint or globally attached at a lower scope, effectively disables a globally attached authorization policy at a higher scope. If the globally attached policy contains any other assertions, in addition to the authorization assertion, those assertions are disabled also.

B.6.6 oracle/no_authorization_component_policy

This policy, when directly attached to a SOA component or globally attached at a lower scope, effectively disables a globally attached authorization policy at a higher scope. If the globally attached policy contains any other assertions, in addition to the authorization assertion, those assertions are disabled also.

B.6.7 oracle/no_addressing_policy

This policy, when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached WS Addressing policy at a higher scope.

B.6.8 oracle/no_mtom_policy

This policy, when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached WS MTOM policy at a higher scope.

B.6.9 oracle/no_wsrm_policy

This policy, when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached Web Services Reliable Messaging policy at a higher scope.

C

Predefined Assertion Templates

This appendix describes the predefined assertion templates that you can use to construct your policies or copy to create new policies.

> **Note:** Oracle recommends that you do not edit the predefined assertion templates so that you will always have a known set of valid templates. You can, however, create a new assertion template from a predefined assertion template, or configure the attributes in an assertion after you have added it to a policy. For information about managing the assertion templates and adding them to policies, see "Managing Policy Assertion Templates" on page 7-7.

This chapter contains the following sections:

- Security Assertion Templates
- Management Assertion Templates
- No Behavior Assertion Templates

C.1 Security Assertion Templates

The following sections describe the security assertion templates in more detail.

- Authentication Only Assertion Templates
- Message-Protection Only Assertion Templates
- Message Protection and Authentication Assertion Templates
- WS-Trust Assertion Templates
- Authorization Assertion Templates
- Oracle Entitlements Server (OES) Integration Templates
- Supported Algorithm Suites
- Message Signing and Encryption Settings for Request, Response, and Fault Messages

You can jump to a specific assertion template description using the following links (listed alphabetically):

- oracle/binding_authorization_template
- oracle/binding_permission_authorization_template
- oracle/component_authorization_template

- oracle/component_permission_authorization_template
- oracle/http_oam_token_service_template
- oracle/http_saml20_token_bearer_client_template or oracle/http_saml20_token_bearer_service_template
- oracle/http_spnego_token_client_template or oracle/http_spnego_token_service_template
- oracle/http_jwt_token_client_template or oracle/http_jwt_token_service_template
- oracle/http_jwt_token_over_ssl_client_template or oracle/http_jwt_token_over_ssl_service_template
- oracle/http_oauth2_token_client_template
- oracle/http_oauth2_token_over_ssl_client_template
- oracle/oauth2_config_client_template
- oracle/security_log_template
- oracle/sts_trust_config_client_template
- oracle/wss_http_token_client_template or oracle/wss_http_token_service_template
- oracle/wss_http_token_over_ssl_client_template or oracle/wss_http_token_over_ssl_service_template
- oracle/wss_saml_token_bearer_over_ssl_client_template or oracle/wss_saml_token_bearer_over_ssl_service_template
- oracle/wss_saml20_token_bearer_over_ssl_client_template or oracle/wss_saml20_token_bearer_over_ssl_service_template
- oracle/wss_saml_token_over_ssl_client_template or oracle/wss_saml_token_over_ssl_service_template
- oracle/wss_saml20_token_over_ssl_client_template or oracle/wss_saml20_token_over_ssl_service_template
- oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template or oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template
- oracle/wss_username_token_over_ssl_client_template or oracle/wss_username_token_over_ssl_service_template
- oracle/wss_username_token_client_template or oracle/wss_username_token_service_template
- oracle/wss_username_token_over_ssl_client_template or oracle/wss_username_token_over_ssl_service_template
- oracle/wss10_message_protection_client_template or oracle/wss10_message_protection_service_template
- oracle/wss10_saml_token_client_template or oracle/wss10_saml_token_service_template
- oracle/wss10_saml20_token_client_template or oracle/wss10_saml20_token_service_template
- oracle/wss10_saml_token_with_message_protection_client_template or oracle/wss10_saml_token_with_message_protection_service_template

- oracle/wss10_saml20_token_with_message_protection_client_template or oracle/wss10_saml20_token_with_message_protection_service_template
- oracle/wss10_username_token_with_message_protection_client_template or oracle/wss10_username_token_with_message_protection_service_template
- oracle/wss10_x509_token_with_message_protection_client_template or oracle/wss10_x509_token_with_message_protection_service_template
- oracle/wss11_kerberos_token_client_template or oracle/wss11_kerberos_token_service_template
- oracle/wss11_kerberos_token_with_message_protection_client_template or oracle/wss11_kerberos_token_with_message_protection_service_template
- oracle/wss11_saml_token_with_message_protection_client_template or oracle/wss11_saml_token_with_message_protection_service_template
- oracle/wss11_saml20_token_with_message_protection_client_template or oracle/wss11_saml20_token_with_message_protection_service_template
- oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template or oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template
- oracle/wss11_sts_issued_saml_with_message_protection_client_template
- oracle/wss11_username_token_with_message_protection_client_template or oracle/wss11_username_token_with_message_protection_service_template
- oracle/wss11_x509_token_with_message_protection_client_template or oracle/wss11_x509_token_with_message_protection_service_template

C.1.1 Authentication Only Assertion Templates

Table C–1 summarizes the assertion templates that enforce authentication only, and indicates whether the token is inserted at the transport layer or SOAP header.

Table C–1 Authentication Only Assertion Templates

Client Template	Service Template	Authentication Transport	Authentication SOAP	Authentication REST	Message Protection Transport	Message Protection SOAP
oracle/http_jwt_token_client_template	oracle/http_jwt_token_service_template	Yes	No	Yes	No	No
N/A	oracle/http_oam_token_service_template	No	No	Yes	No	No
oracle/http_oauth2_token_client_template	N/A	No	No	No	No	No
oracle/oauth2_config_client_template	N/A	No	No	No	No	No
oracle/http_saml20_token_bearer_client_template	oracle/http_saml20_token_bearer_service_template	No	No	Yes	Yes	No
oracle/http_spnego_token_client_template	oracle/http_spnego_token_service_template	No	No	Yes	Yes	No

Table C–1 (Cont.) Authentication Only Assertion Templates

Client Template	Service Template	Authentication Transport	Authentication SOAP	Authentication REST	Message Protection Transport	Message Protection SOAP
oracle/wss_http_token_client_template	oracle/wss_http_token_service_template	Yes	No	No	No	No
oracle/wss_username_token_client_template	oracle/wss_username_token_service_template	No	Yes	No	No	No
oracle/wss10_saml_token_client_template	oracle/wss10_saml_token_service_template	No	Yes	No	No	No
oracle/wss10_saml20_token_client_template	oracle/wss10_saml20_token_service_template	No	Yes	No	No	No
oracle/wss11_kerberos_token_client_template	oracle/wss11_kerberos_token_service_template	No	Yes	No	No	No

C.1.1.1 oracle/http_jwt_token_client_template

The http_jwt_token_client_template assertion template includes a JWT token in the HTTP header. The JWT token is created automatically. The issuer name and subject name are provided either programmatically or declarative through the policy. A policy created using this template can be attached to any HTTP-based client. You can specify the audience restriction condition using the configuration override property.

Settings

Table C–2 lists the settings for the http_jwt_token_client_template assertion template.

Note: This template is also the basis for the http_jwt_token_identity_switch_client_policy, which can also perform dynamic identity switching by propagating a different identity than the one based on authenticated Subject. However, in the assertion content the `subject.precedence` config-override property defaults to `false`.

Table C–2 http_jwt_token_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ■ basic—Client authenticates itself by transmitting the username and password. **Note**: It is recommended that you configure SSL when using basic authentication. For more information, see "Configuring Keystores for SSL" on page 10-36. ■ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ■ custom—**Not supported in this release**. Custom authentication mechanism. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ■ jwt—Client authenticates itself using JWT token. ■ oam—Client authenticates itself using OAM agent. ■ saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. ■ spnego—Client authenticates itself using Kerberos SPNEGO.	`<orasp:auth-header orasp:mechanism="jwt"/>`
Authentication Header—Header Name	Name of the authentication header.	None
Authentication Header—algorithm-suite	Algorithm suite used to sign the JWT token.	`<orasp:auth-header orasp:algorithm-suite="Basic128Sha256Rsa15"/"` **Note:** This is the only supported value. If any value other than this default value is specified, the policy will fail.
Authentication Header—is-signed	Flag that specifies whether the JWT token is signed. The only valid value for JWT policies is: `true`.	`<orasp:auth-header orasp:is-signed="true"/>`
Authentication Header— is encrypted	Flag that specifies whether the JWT token is encrypted.	`<orasp:auth-header orasp:is-encrypted="false"/>`

Configurations

Table C–3 lists the configuration properties and the default settings for the http_jwt_token_client_template assertion template.

Table C–3 http_jwt_token_client_template Configuration Properties

Name	Default Values

audience.uri	Audience restriction. The following conditions are supported:
	- If this property is not set, the service URL is used as the audience URI
	- If this property is set to `NONE` (not case sensitive), then the audience URI is set to null.
	- If this property is set to a value other than `NONE`, then the audience URI is set to this value.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="audience.uri" orawsp:type="string">`
	`  <orawsp:Value/>`
	`</orawsp:Property>`

csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services (OPSS) identity store.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf-key" orawsp:type="string">`
	`   <orawsp:Value>basic.credentials</orawsp:Value>`
	`</orawsp:Property>`

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="csf.map" orawsp:type="string"/>`
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as a `Value` in this property. For example:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

issuer.name	Name of the JWT issuer. The default value is www.oracle.com.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="issuer.name" orawsp:type="string">`
	`  <orawsp:Value>www.oracle.com</orawsp:Value>`
	`</orawsp:Property>`

keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="keystore.sig.csf.key" orawsp:type="string"/>`

Table C–3 (Cont.) http_jwt_token_client_template Configuration Properties

Name	Default Values

propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="propagate.identity.context" orawsp:type="string">`
	`<orawsp:Value/>`

reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope.
	The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1.
	For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="reference.priority" orawsp:type="string"/>`

subject.precedence	Property that specifies the location from which the subject used to create the JWT token should be obtained.
	If `subject.precedence` is set to `true`, the user name to create the JWT token is obtained only from the authenticated Subject. If `subject.precedence` is set to `false`, the user name to create the JWT token is obtained only from the csf-key username property.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="subject.precedence" orawsp:type="string">`
	`  <orawsp:Value>true</orawsp:Value>`
	`</orawsp:Property>`
	Note: When configuring a http_jwt_token_bearer_identity_switch_client_policy, the `subject.precedence` value defaults to `false` for dynamic identity switching.

Table C–3 (Cont.) http_jwt_token_client_template Configuration Properties

Name	Default Values

user.attributes	List of user attributes for the authenticated user to be included in the JWT token.
	Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the JWT token.
	Requires that the Subject is available and `subject.precedence` is set to true.
	A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create JWT claims.
	The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.
	If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`orawsp:name="user.attributes" orawsp:type="string"/>`

user.roles.include	User roles to be included in the JWT token. If set to `true`, the authenticated user roles are included in the JWT token as private claims. The default is `false`.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`orawsp:name="user.roles.include" orawsp:type="string">`
	`<orawsp:Value>false</orawsp:Value>`
	`</orawsp:Property>`

user.tenant.name	Reserved for use with Oracle Cloud.

C.1.1.2 oracle/http_jwt_token_service_template

This oracle/http_jwt_token_service_template authenticates users using the credentials provided in the JWT token in the HTTP header.

Settings

The settings for the http_jwt_token_service_template assertion template are identical to the client version of the assertion template. See Table C–2 for information about the settings.

Configurations

Table C–4 lists the configuration properties and the default settings for the http_jwt_token_service_template assertion template.

Table C–4 http_jwt_token_service_template Configuration Properties

Name	Default Values

trusted.issuers	A comma-separated list of trusted issuers for an application that will override the trusted issuers defined at the domain level.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="saml.trusted.issuers" orawsp:type="string">`
	`  <orawsp:Value/>`
	`</orawsp:Property>`

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="csf.map" orawsp:type="string"/>`
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as a `Value` in this property. For example:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="keystore.sig.csf.key" orawsp:type="string"/>`

reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope.
	The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1.
	For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="reference.priority" orawsp:type="string"/>`

propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="propagate.identity.context" orawsp:type="string">`
	`<orawsp:Value/>`

C.1.1.3 oracle/http_oam_token_service_template

The http_oam_token_service_template assertion template verifies that OAM agent has authenticated the user and has established an identity. This policy can be applied to any HTTP-based endpoint.

Settings

Table C–5 lists the settings for the http_oam_token_service_template assertion template.

Table C–5 http_oam_token_service_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ▪ basic—Client authenticates itself by transmitting the username and password. **Note**: It is recommended that you configure SSL when using basic authentication. For more information, see "Configuring Keystores for SSL" on page 10-36. ▪ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ▪ custom—**Not supported in this release**. Custom authentication mechanism. ▪ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ▪ jwt—Client authenticates itself using JWT token. ▪ oam—Client authenticates itself using OAM agent. ▪ saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. ▪ spnego—Client authenticates itself using Kerberos SPNEGO.	`<orasp:auth-header orasp:mechanism="oam"/>`
Authentication Header—Header Name	Name of the authentication header.	None

Configurations

Table C–6 lists the default configuration properties for the http_oam_token_service_template assertion template.

Table C–6 http_oam_token_service_template Configuration Properties

Name	Description
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35. Default setting: `<orawsp:Property orawsp:contentType="optional" orawsp:name="reference.priority" orawsp:type="string"/>`

C.1.1.4 oracle/http_oauth2_token_client_template

The http_oauth2_token_client_template assertion template is the HTTP binding level template for OAuth2 token authentication.

Settings

Table C–7 lists the settings for the http_oauth2_token_client_template assertion template.

Table C–7 http_oauth2_token_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ■ basic—Client authenticates itself by transmitting the username and password. **Note**: It is recommended that you configure SSL when using basic authentication. For more information, see "Configuring Keystores for SSL" on page 10-36. ■ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ■ custom—**Not supported in this release**. Custom authentication mechanism. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ■ jwt—Client authenticates itself using JWT token. ■ oam—Client authenticates itself using OAM agent. ■ oauth2—Client authenticates using OAuth2 framework. ■ saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. ■ spnego—Client authenticates itself using Kerberos SPNEGO.	`<orasp:auth-header orasp:mechanism="oauth2"/>`
Authentication Header—Header Name	Name of the authentication header.	None
Authentication Header—is-signed	Flag that specifies whether the token is signed.	`<orasp:auth-header orasp:is-signed="false"/>`
Authentication Header— is encrypted	Flag that specifies whether the token is encrypted.	`<orasp:auth-header orasp:is-encrypted="false"/>`

Configurations

Table C–8 lists the default configuration properties for the http_oauth2_token_client_template assertion template.

Table C–8 http_oauth2_token_client_template Configuration Properties

Name	Description

audience.uri	Audience restriction. The following conditions are supported:
	- If this property is not set, the service URL is used as the audience URI
	- If this property is set to `NONE` (not case sensitive), then the audience URI is set to null.
	- If this property is set to a value other than `NONE`, then the audience URI is set to this value.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="audience.uri"`
	`orawsp:type="string">`
	`<orawsp:Value/>`
	`<orawsp:DefaultValue>NONE</orawsp:DefaultValue>`

authz.code	Optional property for passing the authorization code for the 3-legged OAuth2 use case. (Not supported in this release.)
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="authz.code"`
	`orawsp:type="string">`
	`<orawsp:Value/>`

csf-key	Credential store key that maps to a user name and password in the Oracle Platform Security Services (OPSS) identity store.
	Default setting:
	`<orawsp:Property orawsp:type="string"`
	` orawsp:contentType="optional" orawsp:name="csf-key">`
	`<orawsp:Value/>`

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="csf.map" orawsp:type="string"/>`
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as a `Value` in this property. For example:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–8 (Cont.) http_oauth2_token_client_template Configuration Properties

Name	Description

federated.client.token	Optional property which, by default, specifies that a JWT token is generated for the client using the values of the oauth2.client.csf.key and keystore.sig.csf.key properties.
	If set to false, oauth2.client.csf.key is used to generate an Authorization header sent in the client request to the OAuth server.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="federated.client.token"`
	`orawsp:type="boolean">`
	`<orawsp:Value/>`
	`<orawsp:DefaultValue>true</orawsp:DefaultValue>`

include.certificate	When true, the signature certificate and the trusted certificate chain (for CA-issued certificates) are included in JWT token claim. This increases the size of the JWT token, but you do not need to then import the certificate and certificate chain into the service side keystore.
	When false, only the thumbprint and alias of the certificate are included in the JWT token.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="include.certificate"`
	`orawsp:type="string">`
	`<orawsp:Value/>`
	`<orawsp:DefaultValue>false</orawsp:DefaultValue>`
	`</orawsp:Property>`

issuer.name	Optional property that specifies the issuer name used for the locally-generated JWT token (iss:claim). By default it is www.oracle.com.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="issuer.name"`
	`orawsp:type="string">`
	`<orawsp:Value/>`
	`<orawsp:DefaultValue>www.oracle.com</orawsp:DefaultValue>`

keystore.sig.csf.key	Optional property that specifies the tenant key from the Oracle WSM keystore for signing the locally-created JWT token.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="keystore.sig.csf.key"`
	`orawsp:type="string">`
	`<orawsp:Value/>`

Table C–8 (Cont.) http_oauth2_token_client_template Configuration Properties

Name	Description

oauth2.client.csf.key	Required property that specifies the key to use to obtain the client username and password.
	The value of oauth2.client.csf.key must match the client ID and secret expected by the client profile, as described in "Understanding OAuth Client Profiles Configuration" in *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.
	If `federated.client.token` is set to false, oauth2.client.csf.key is used to generate an Authorization header sent in the client request to the OAuth server.
	If you override oauth2.client.csf.key, that value is used. Otherwise, the value of oauth2.client.csf.key in oauth2_config_client_policy is used.
	Default setting:
	`<orawsp:Property orawsp:type="string"`
	`orawsp:contentType="required"`
	`orawsp:name="oauth2.client.csf.key">`
	`<orawsp:Value/>`
	`<orawsp:DefaultValue>NONE</orawsp:DefaultValue>`
	`</orawsp:Property>`

oracle.oauth2.service	Optional property that specifies how the default behavior of token issuer and scope are determined. When true, the client ID is used as the issuer of the user and client JWT token for the OAuth2 server. In this case, the value for `issuer.name` is ignored. When false, the issuer is determined by `issuer.name` with the default value of "www.oracle.com".

propagate.identity.context	Optional property that specifies whether the identity context information is propagated as claims in the JWT token.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="propagate.identity.context"`
	`orawsp:type="string">`
	`<orawsp:Value/>`

redirect.uri	Optional property that specifies the redirect URIs that the OAuth server will use to redirect the user-agent to the client once access is granted or denied.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="redirect.uri"`
	`orawsp:type="string">`
	`<orawsp:Value/>`

Table C–8 (Cont.) http_oauth2_token_client_template Configuration Properties

Name	Description

reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope.
	The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1.
	For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="reference.priority"`
	`  orawsp:type="string"/>`

scope	Optional property that specifies the scope (as-is) of the OAuth2 request. If present, the scope is included in the OAuth2 token request with the value.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="scope" orawsp:type="string">`
	`<orawsp:Value/>`
	The scope depends on the value of the `oracle.oauth2.service` property:
	■ If `oracle.oauth2.service` is false (the default), the `scope` property determines the scope.
	■ If `oracle.oauth2.service` is true and `scope` has no value, (the default), the protocol, host and port (if available) are obtained from the service URL and used.

subject.precedence	Property that specifies the location from which the subject used to create the JWT token should be obtained.
	As described in Table 10–2, " User Credential, Subject, and Access Token":
	■ If `subject.precedence` is set to `true`, the user name to create the JWT token is obtained only from the authenticated subject.
	■ If `subject.precedence` is set to `false`, the user name to create the JWT token is obtained only from the `csf-key` property.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="subject.precedence" orawsp:type="string">`
	`  <orawsp:Value>true</orawsp:Value>`
	`</orawsp:Property>`

Table C–8 (Cont.) http_oauth2_token_client_template Configuration Properties

Name	Description

time.in.millis	Support standard NumericDate (seconds after Epoch as unit for values in exp (Expiry) and iat (Issued AT) claims in JWT token.
	If true, then milliseconds after Epoch is used. Otherwise, seconds after Epoch is used.
	Default setting:
	`<orawsp:Property orawsp:type="boolean"`
	` orawsp:contentType="optional"`
	` orawsp:name="time.in.millis">`
	`<orawsp:Value/>`
	`<orawsp:DefaultValue>true</orawsp:DefaultValue>`
	`</orawsp:Property>`

user.attributes	Optional property that specifies whether user attributes are inserted as claims in JWT token.
	Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the JWT token.
	Requires that the Subject is available and `subject.precedence` is set to true.
	A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create JWT claims.
	The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.
	If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="user.attributes"`
	`orawsp:type="string">`
	`<orawsp:Value/>`

user.roles.include	Optional property that specifies whether the user roles from the subject are included in the JWT token as claims. If set to true, the authenticated user roles are included in the JWT token as private claims.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="user.roles.include"`
	`orawsp:type="boolean">`
	`<orawsp:Value/>`
	`<orawsp:DefaultValue>false</orawsp:DefaultValue>`

user.tenant.name	Reserved for use with Oracle Cloud.

C.1.1.5 oracle/oauth2_config_client_template

The oauth2_config_client_template assertion template provides OAuth2 information that is used to invoke the OAuth2 server for obtaining an access token.

Settings

Table C–9 lists the settings for the oauth2_config_client_template assertion template.

Table C–9 oauth2_config_client_template Settings

Name	Description	Default Value
token-uri	Required property that specifies the token endpoint of the OAuth2 server.	`orasp:token-uri="http://host:port/tokens"`

Configurations

Table C–10 lists the default configuration properties for the oauth2_config_client_template assertion template.

Table C–10 oauth2_config_client_template Configuration Properties

Name	Description

oauth2.client.csf.key	Required property that specifies the key to use to obtain the client username and password.
	The value of oauth2.client.csf.key must match the client ID and secret expected by the client profile, as described in "Understanding OAuth Client Profiles Configuration" in *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.
	Default setting:
	`<orawsp:Property orawsp:type="string"`
	`orawsp:contentType="required"\`
	`orawsp:name="oauth2.client.csf.key">`
	`<orawsp:Value/>`
	`<orawsp:DefaultValue>basic.client.credentials</orawsp:Defau`
	`ltValue>`
	`</orawsp:Property>`

role	SOAP role.
	Default setting:
	`<orawsp:Property orawsp:contentType="constant"`
	`  orawsp:name="role" orawsp:type="string">`
	`  <orawsp:DefaultValue>`
	`    ultimateReceiver`
	`  </orawsp:DefaultValue>`
	`</orawsp:Property>`

reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope.
	The value of reference.priority can be any number between(-2^{31}) and $(2^{31} - 1)$. The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1.
	For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="reference.priority" orawsp:type="string"/>`

token.uri	Optional property to override the token-uri value.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`orawsp:name="token.uri"`
	` orawsp:type="string">`
	`<orawsp:Value/>`
	`<orawsp:DefaultValue>http://host:port/tokens`
	`</orawsp:DefaultValue>`
	`</orawsp:Property>`

C.1.1.6 oracle/http_saml20_token_bearer_client_template

The http_saml20_token_bearer_client template assertion template includes SAML 2,0 tokens in outbound SOAP request messages. The SAML token with confirmation method [*Bearer*] is created automatically.

Settings

Table C–11 lists the settings for the http_saml20_token_bearer_client_template assertion template.

Table C–11 http_saml20_token_bearer_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ■ basic—Client authenticates itself by transmitting the username and password. **Note**: It is recommended that you configure SSL when using basic authentication. For more information, see "Configuring Keystores for SSL" on page 10-36. ■ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ■ custom—**Not supported in this release**. Custom authentication mechanism. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ■ jwt—Client authenticates itself using JWT token. ■ oam—Client authenticates itself using OAM agent. ■ saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. ■ spnego—Client authenticates itself using Kerberos SPNEGO.	`<orasp:auth-header orasp:mechanism="saml20-bearer"/>`
Authentication Header—Header Name	Name of the authentication header.	None

Configurations

Table C–12 lists the configuration properties and the default settings for the http_saml20_token_bearer_client_template assertion template.

Table C–12 http_saml20_token_bearer_client_template Configuration Properties

Name	Default Values

user.attributes	User attributes related to the principal of the SAML token.
	Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion.
	Requires that the Subject is available and `subject.precedence` is set to true.
	A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement.
	The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59.
	If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="user.attributes" orawsp:type="string"/>`

saml.issuer.name	Issuer URI.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="saml.issuer.name" orawsp:type="string">`
	`  <orawsp:Value>www.oracle.com</orawsp:Value>`
	`</orawsp:Property>`

user.roles.include	User roles to be included.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="user.roles.include" orawsp:type="string">`
	`  <orawsp:Value>false</orawsp:Value>`
	`</orawsp:Property>`

csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="csf-key" orawsp:type="string">`
	`  <orawsp:Value>basic.credentials</orawsp:Value>`
	`</orawsp:Property>`

Table C–12 (Cont.) http_saml20_token_bearer_client_template Configuration Properties

Name	Default Values

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="csf.map" orawsp:type="string"/>`
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as a `Value` in this property. For example:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject.
	If `subject.precedence` is true, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="subject.precedence" orawsp:type="string">`
	`  <orawsp:Value>true</orawsp:Value>`
	`</orawsp:Property>`

saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards:
	■ `*` in any location.
	■ `/*` at the end of the URI.
	■ `.*` at the end of the URI.
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="saml.audience.uri" orawsp:type="string">`
	`  <orawsp:Value/>`
	`</orawsp:Property>`

keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. This key is used when generating the enveloping signature, as specified using `saml.envelope.signature.required` flag.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="keystore.sig.csf.key" orawsp:type="string"/>`

Table C–12 (Cont.) http_saml20_token_bearer_client_template Configuration Properties

Name	Default Values

saml.envelope.signature.required	Flag that specifies whether the bearer token is signed using the domain signature key. You can override the domain signature key using the private signature key configured using `keystore.sig.csf.key`.
	Set this flag `false` (in both client and service policy) to have the bearer token be unsigned.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="saml.enveloped.signature.required"`
	`  orawsp:type="boolean">`
	`  <orawsp:Value>true</orawsp:Value>`
	`</orawsp:Property>`

reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope.
	The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1.
	For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="reference.priority" orawsp:type="string"/>`

propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="propagate.identity.context" orawsp:type="string">`
	`<orawsp:Value/>`

C.1.1.7 oracle/http_saml20_token_bearer_service_template

The http_saml20_token_bearer_service_template assertion template authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

Settings

The settings for the http_saml20_token_bearer_service_template assertion template are identical to the client version of the assertion template. See Table C–11 for information about the settings.

Configurations

Table C–13 lists the configuration properties and the default settings for the http_saml20_token_bearer_service_template assertion template.

Table C–13 http_saml20_token_bearer_service_template Configuration Properties

Name	Default Values

saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="saml.trusted.issuers" orawsp:type="string">`
	`  <orawsp:Value/>`
	`</orawsp:Property>`

saml.envelope.signature.required	Flag that specifies whether the bearer token is signed using the domain signature key. You can override the domain signature key using the private signature key configured using `keystore.sig.csf.key`.
	Set this flag `false` (in both client and service policy) to have the bearer token be unsigned.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="saml.enveloped.signature.required"`
	`  orawsp:type="boolean">`
	`  <orawsp:Value>true</orawsp:Value>`
	`</orawsp:Property>`

reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope.
	The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1.
	For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="reference.priority" orawsp:type="string"/>`

propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="propagate.identity.context" orawsp:type="string">`
	`<orawsp:Value/>`

C.1.1.8 oracle/http_spnego_token_client_template

The http_spnego_token_client_template assertion template provides authentication using a Kerberos token and the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) protocol.

Settings

Table C–14 lists the settings for the http_spnego_token_client_template assertion template.

Table C–14 http_spnego_token_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: - basic—Client authenticates itself by transmitting the username and password. **Note**: It is recommended that you configure SSL when using basic authentication. For more information, see "Configuring Keystores for SSL" on page 10-36. - cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. - custom—**Not supported in this release**. Custom authentication mechanism. - digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. - jwt—Client authenticates itself using JWT token. - oam—Client authenticates itself using OAM agent. - saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. - spnego—Client authenticates itself using Kerberos SPNEGO.	`<orasp:auth-header orasp:mechanism="spnego"/>`
Authentication Header—Header Name	Name of the authentication header.	None

Configurations

Table C–15 lists the default configuration properties for the http_spnego_token_client_template assertion template.

Table C–15 http_spnego_token_client_template Configuration Properties

Name	Default Values

service.principal.name	Kerberos principal name that identifies the service.
	Default setting:
	`<orawsp:Property orawsp:name="service.principal.name"`
	`  orawsp:type="string">`
	`  <orawsp:Value>HOST/localhost@EXAMPLE.COM</orawsp:Value>`
	`</orawsp:Property>`

keytab.location	Location of the client's keytab file.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="keytab.location" orawsp:type="string">`
	`  <orawsp:Value/>`
	`</orawsp:Property>`

caller.principal.name	Client's principal name as generated using the `ktpass` command and mapped to the username for which the kerberos token should be generated. Use the following format: `<username>@<REALM NAME>`.
	Note: `keytab.location` and `caller.principal.name` are required for propagating client identity for Java EE applications.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="caller.principal.name"`
	`  orawsp:type="string">`
	`  <orawsp:Value/>`
	`</orawsp:Property>`

role	SOAP role.
	Default setting:
	`<orawsp:Property orawsp:contentType="constant"`
	`  orawsp:name="role" orawsp:type="string">`
	`  <orawsp:DefaultValue>`
	`    ultimateReceiver`
	`  </orawsp:DefaultValue>`
	`</orawsp:Property>`

reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope.
	The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1.
	For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="reference.priority" orawsp:type="string"/>`

C.1.1.9 oracle/http_spnego_token_service_template

This http_spnego_token_service_template assertion template provides authentication using a Kerberos token and the SPNEGO protocol.

Settings

The settings for the http_spnego_token_service_template assertion template are identical to the client version of the assertion template. See Table C–14 for information about the settings.

Configurations

Table C–16 lists the default configuration properties for the http_spnego_token_service_template assertion template.

Table C–16 http_spnego_token_service_template Configuration Properties

Name	Default Values
role	SOAP role. Default setting: `<orawsp:Property orawsp:contentType="constant" orawsp:name="role" orawsp:type="string"> <orawsp:DefaultValue> ultimateReceiver </orawsp:DefaultValue> </orawsp:Property>`
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35. Default setting: `<orawsp:Property orawsp:contentType="optional" orawsp:name="reference.priority" orawsp:type="string"/>`

C.1.1.10 oracle/wss_http_token_client_template

The wss_http_token_client_template assertion template includes username and password credentials in the HTTP header. You can control whether one-way or two-way authentication is required.

Settings

Table C–17 lists the settings for the wss_http_token_client_template assertion template.

Table C–17 wss_http_token_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ■ basic—Client authenticates itself by transmitting the username and password. **Note**: It is recommended that you configure SSL when using basic authentication. For more information, see "Configuring Keystores for SSL" on page 10-36. ■ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ■ custom—**Not supported in this release**. Custom authentication mechanism. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ■ jwt—Client authenticates itself using JWT token. ■ oam—Client authenticates itself using OAM agent. ■ saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. ■ spnego—Client authenticates itself using Kerberos SPNEGO.	basic
Authentication Header—Header Name	Name of the authentication header.	None
Transport Security	Flag that specifies whether SSL is enabled.	Enabled
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: ■ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ■ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	Disabled
Transport Security—Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Disabled

Configurations

Table C–18 lists the configuration properties and the default settings for the wss_http_token_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–18 wss_http_token_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—Not set ■ Default—basic.credentials ■ ContentType—Required ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
user.tenant.name	Reserved for use with Oracle Cloud.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.1.11 oracle/wss_http_token_service_template

The wss_http_token_service_template assertion template uses the credentials in the HTTP header to authenticate users against the Oracle Platform Security Services identity store. You can control whether one-way or two-way authentication is required.

Settings

The settings for the wss_http_token_service_template are identical to those for the client version of the assertion template. See Table C–17 for information on the settings.

Configurations

Table C–19 lists the configuration properties and the default settings for the wss_http_token_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–19 wss_http_token_service_template Configurations

Name	Description
realm	HTTP Realm. Default settings: ■ Value—Not set ■ Default—owsm ■ ContentType—Constant ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.1.12 oracle/wss_username_token_client_template

The wss_username_token_client_template assertion template includes authentication with username and password credentials in the WS-Security UsernameToken header.

The assertion supports three types of password credentials: plain text, digest, and no password.

> **Note:** Digest passwords are not supported in this release.
>
> Policies created using this template are not secure; it transmits the password in clear text. You should use this assertion in low security situations only, or when you know that the transport is protected using some other mechanism. Alternatively, consider using the SSL version of this assertion, oracle/wss_username_token_over_ssl_client_template.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token.

Settings

Table C–20 lists the settings for the wss_username_token_client_template assertion template.

Table C–20 wss_username_token_client_template Settings

Name	Description	Default Value
Password Type	Type of password required. Valid values are: ■ none—No password. ■ plaintext—Password in clear text. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. **Note**: The plaintext type is not recommended when the token propagation occurs on an unsecure channel. However, if SSL is being used as the transport channel to secure a point-to-point connection between client and server, the plaintext type can be used as the channel takes care of protecting the password.	plaintext
Nonce Required	Flag that specifies whether a nonce must be included with the username to prevent replay attacks. **Notes**: ■ If Password Type is set to digest, then this attribute must be set to true. Otherwise, the policy to which it is attached will not validate. ■ If Creation Time Required is set to true, than this attribute must be set to true. Otherwise, nonce will be cached forever to prevent replay attacks.	False
Creation Time Required	Flag that specifies whether a time stamp for the creation of the username token is required. **Notes**: ■ If Password Type is set to digest, then this attribute must be set to true. Otherwise, the policy to which it is attached will not validate. ■ If Nonce Required is set to true, than this attribute must be set to true. Otherwise, nonce will be cached forever to prevent replay attacks.	False

Configurations

Table C–21 lists the configuration properties and the default settings for the wss_username_token_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–21 wss_username_token_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ▪ Value—Not set ▪ Default—basic.credentials ▪ ContentType—Required ▪ Description—Not set

Table C–21 (Cont.) wss_username_token_client_template Configurations

Name	Description
role	SOAP role. Default settings: - Value—Not set - Default—ultimateReceiver - ContentType—Constant - Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
user.tenant.name	Reserved for use with Oracle Cloud.

C.1.1.13 oracle/wss_username_token_service_template

The wss_username_token_service_template assertion template enforces authentication with username and password credentials in the WS-Security UsernameToken SOAP header. The assertion supports three types of password credentials: plain text, digest, and no password.

> **Note:** Digest passwords are not supported in this release.
>
> Policies created using this template are not secure; it transmits the password in clear text. You should use this assertion in low security situations only, or when you know that the transport is protected using some other mechanism. Alternatively, consider using the SSL version of this assertion, oracle/wss_username_token_over_ssl_service_template.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token.

Settings

The settings for the wss_username_token_service_template are identical to the client version of the assertion template. See Table C–20 for information on the settings.

Configurations

Table C–22 lists the configuration properties and the default settings for the wss_username_token_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–22 wss_username_token_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.1.14 oracle/wss10_saml_token_client_template

The wss10_saml_token_client_template assertion template includes SAML tokens in outbound SOAP request messages. The SAML token is created automatically.

Settings

Table C–23 lists the settings for the wss10_saml_token_client_template assertion template.

Table C–23 wss10_saml_token_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: ■ sender-vouches—Uses the Sender Vouches SAML token for authentication.	sender-vouches
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is true, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName	unspecified

Configurations

Table C–24 lists the configuration properties and the default settings for the wss10_saml_token_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–24 wss10_saml_token_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to true. Default settings: ■ Value—Not set ■ Default—Not set. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
user.roles.include	User roles to be included. Default settings: ■ Value—Not set ■ Default—false ■ ContentType—Optional ■ Description—Not set
saml.issuer.name	Issuer URI. Default settings: ■ Value—Not set ■ Default—www.oracle.com ■ ContentType—Optional ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–24 (Cont.) wss10_saml_token_client_template Configurations

Name	Description
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is true, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—null ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.1.15 oracle/wss10_saml_token_service_template

The wss10_saml_token_service_template assertion template authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.

Settings

The settings for the wss10_saml_token_service_template are identical to the client version of the assertion, with the exception that Name Identifier Format is not present. See Table C–23 for information on the settings.

Configurations

Table C–25 lists the configuration properties and the default settings for the wss10_saml_token_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–25 wss10_saml_token_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set

Table C–25 (Cont.) wss10_saml_token_service_template Configurations

Name	Description
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: ■ Value—Not set ■ Default—null ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.1.16 oracle/wss10_saml20_token_client_template

The wss10_saml20_token_client_template assertion template includes SAML tokens in outbound SOAP request messages. The SAML token is created automatically.

Settings

Table C–26 lists the settings for the wss10_saml20_token_client_template assertion template.

Table C–26 wss10_saml20_token_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is 2.0.	2.0
Confirmation Type	Confirmation type. The only valid value is: ■ sender-vouches—Uses the Sender Vouches SAML token for authentication.	sender-vouches
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is true, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName ■ kerberos	unspecified

Configurations

Table C–27 lists the configuration properties and the default settings for the wss10_saml20_token_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–27 wss10_saml20_token_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to true. Default settings: ■ Value—Not set ■ Default—Not set. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
user.roles.include	User roles to be included. Default settings: ■ Value—Not set ■ Default—false ■ ContentType—Optional ■ Description—Not set
saml.issuer.name	Issuer URI. Default settings: ■ Value—Not set ■ Default—www.oracle.com ■ ContentType—Optional ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–27 (Cont.) wss10_saml20_token_client_template Configurations

Name	Description
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is true, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—null ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and ($2^{31} - 1$). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.1.17 oracle/wss10_saml20_token_service_template

The wss10_saml20_token_service_template assertion template authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.

Settings

The settings for the wss10_saml20_token_service_template are similar to the client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–26 for information on the settings.

Configurations

Table C–28 lists the configuration properties and the default settings for the wss10_saml20_token_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–28 wss10_saml20_token_service_template Configurations

Name	Description
role	SOAP role. Default settings: ▪ Value—Not set ▪ Default—ultimateReceiver ▪ ContentType—Constant ▪ Description—Not set

Table C–28 (Cont.) wss10_saml20_token_service_template Configurations

Name	Description
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: ■ Value—Not set ■ Default—null ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.1.18 oracle/wss11_kerberos_token_client_template

The wss11_kerberos_token_client_template assertion template includes a Kerberos token in the WS-Security header in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

Settings

Table C–29 lists the settings for the wss11_kerberos_token_client_template assertion template.

Table C–29 wss11_kerberos_token_client_template Settings

Name	Description	Default Value
Kerberos Token Type	Type of Kerberos token. The only valid value is: gss-apreq-v5 (Kerberos Version 5 GSS-API).	gss-apreq-v5

Configurations

Table C–30 lists the configuration properties and the default settings for the wss11_kerberos_token_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–30 wss11_kerberos_token_client_template Configurations

Name	Description
service.principal.name	Kerberos principal name that identifies the service. Default settings: ■ Value—Not set ■ Default—HOST/localhost@EXAMPLE.COM ■ ContentType—Required ■ Description—Not set
keytab.location	Location of the client's keytab file. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
caller.principal.name	Client's principal name as generated using the `ktpass` command and mapped to the username for which the kerberos token should be generated. Use the following format: `<username>@<REALM NAME>`. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set **Note:** `keytab.location` and `caller.principal.name` are required for propagating client identity for Java EE applications.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.1.19 oracle/wss11_kerberos_token_service_template

The wss11_kerberos_token_service_template assertion template enforces in accordance with the WS-Security Kerberos Token Profile v1.1 standard. It extracts the Kerberos token from the SOAP header and authenticates the user. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services.

Settings

The settings for the wss11_keberos_token_service_template are identical to the client version of the assertion template. See Table C–29 for information on the settings.

Configurations

Table C–31 lists the configuration properties and the default settings for the wss11_kerberos_token_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–31 wss11_kerberos_token_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.1.20 oracle/wss_saml_token_bearer_client_template

The wss_saml_token_bearer_client template assertion template includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method [*Bearer*] is created automatically.

Settings

Table C–32 lists the settings for the wss_saml_token_bearer_client_template assertion template.

Note: This template is also the basis for the wss_saml_token_bearer_identity_switch_client_policy, which can perform dynamic identity switching by propagating a different identity than the one based on authenticated Subject. However, in the assertion content the `subject.precedence` config-override property defaults to `false`.

Table C–32 oracle/wss_saml_token_bearer_client_template Settings

Name	Description	Default Values
Version	SAML version. The only valid value is: 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: bearer.	bearer
Is Signed	Flag that specifies whether the SAML token is signed.	False
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is true, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ▪ unspecified ▪ emailAddress ▪ X509SubjectName ▪ WindowsDomainQualifiedName	unspecified

Configurations

Table C–33 lists the configuration properties and the default settings for the wss_saml_token_bearer_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–33 wss_saml_token_bearer_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to true. Default settings: ■ Value—Not set ■ Default—Null. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
user.roles.include	User roles to be included. Default settings: ■ Value—Not set ■ Default—false ■ ContentType—Optional ■ Description—Not set
saml.issuer.name	Issuer URI. Default settings: ■ Value—Not set ■ Default—www.oracle.com ■ ContentType—Optional ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–33 (Cont.) wss_saml_token_bearer_client_template Configurations

Name	Description
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is true, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—true ■ ContentType—Optional ■ Description—Not set **Note:** When configuring a wss_saml_token_bearer_identity_switch_client_policy, the `subject.precedence` value defaults to `false` for dynamic identity switching.
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—null ■ ContentType—Optional ■ Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–33 (Cont.) wss_saml_token_bearer_client_template Configurations

Name	Description
saml.envelope.signature.required	Flag that specifies whether the bearer token is signed using the domain signature key. You can override the domain signature key using the private signature key configured using `keystore.sig.csf.key`. Set this flag `false` (in both client and service policy) to have the bearer token be unsigned. Default setting: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.
user.tenant.name	Reserved for use with Oracle Cloud.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.2 Message-Protection Only Assertion Templates

Table C–34 summarizes the assertion templates that enforce message protection only, and indicates whether the token is inserted at the transport layer or SOAP header.

Table C–34 Message-Protection Only Assertion Templates

Client Template	Service Template	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss10_message_protection_client_template	oracle/wss10_message_protection_service_template	No	No	No	Yes
oracle/wss11_message_protection_client_template	oracle/wss11_message_protection_service_template	No	No	No	Yes

C.1.2.1 oracle/wss10_message_protection_client_template

The wss10_message_protection_client_template assertion template provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

Settings

Table C–35 lists the settings for the wss10_message_protection_client_template assertion template.

Table C–35 wss10_message_protection_client_template Settings

Name	Description	Default Value
X509 Token		
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ▪ direct—X.509 Token is included in the request. ▪ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ▪ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when signing the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–36 lists the configuration properties and the default settings for the wss10_message_protection_client_template assertion template. For details about the

configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–36 wss10_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ▪ Value—Not set ▪ Default—orakey ▪ ContentType—Required ▪ Description—Not set
role	SOAP role. Default settings: ▪ Value—Not set ▪ Default—ultimateReceiver ▪ ContentType—Constant ▪ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set

Table C–36 (Cont.) wss10_message_protection_client_template Configurations

Name	Description
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.2.2 oracle/wss10_message_protection_service_template

The wss10_message_protection_service_template assertion template provides message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

Settings

The settings for the wss10_message_protection_service_template are identical to the client version of the assertion template. See Table C–35 for information on the settings.

Configurations

Table C–37 lists the configuration properties and the default settings for the wss10_message_protection_service_template assertion template. For details about the

configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–37 wss10_message_protection_service_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–37 (Cont.) wss10_message_protection_service_template Configurations

Name	Description
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.2.3 oracle/wss11_message_protection_client_template

The wss11_message_protection_client_template assertion template provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

Settings

Table C–38 lists the settings for the wss11_message_protection_client_template assertion template.

Table C–38 wss11_message_protection_client_template Settings

Name	Description	Default Value
X509 Token		
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead.	thumbprint
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	Enabled
Derived Keys	Flag that specifies whether derived keys should be used.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–39 lists the configuration properties and the default settings for the wss11_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–39 wss11_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—Not set ■ Default—orakey ■ ContentType—Required ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–39 (Cont.) wss11_message_protection_client_template Configurations

Name	Description
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.2.4 oracle/wss11_message_protection_service_template

The wss11_message_protection_service_template assertion template enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

Settings

The settings for the wss11_message_protection_service_template are identical to the client version of the assertion template. See Table C–38 for information on the settings.

Configurations

Table C–40 lists the configuration properties and the default settings for the wss11_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–40 wss11_message_protection_service_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–40 (Cont.) wss11_message_protection_service_template Configurations

Name	Description
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.3 Message Protection and Authentication Assertion Templates

Table C–41 summarizes the assertion templates that enforce both message protection and authentication, and indicates whether the token is inserted at the transport layer or SOAP header.

Table C–41 Message Protection and Authentication Assertion Templates

Client Template	Service Template	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss_http_token_over_ssl_client_template	oracle/wss_http_token_over_ssl_service_template	Yes	No	Yes	No
oracle/wss_saml_token_bearer_client_template	oracle/wss_saml_token_bearer_service_template	No	Yes	Yes	No
oracle/wss_saml_token_bearer_over_ssl_client_template	oracle/wss_saml_token_bearer_over_ssl_service_template	No	Yes	Yes	No
oracle/wss_saml20_token_bearer_over_ssl_client_template	oracle/wss_saml20_token_bearer_over_ssl_service_template	No	Yes	Yes	No
oracle/wss_saml_token_over_ssl_client_template	oracle/wss_saml_token_over_ssl_service_template	No	Yes	Yes	No
oracle/wss_saml20_token_over_ssl_client_template	oracle/wss_saml20_token_over_ssl_service_template	No	Yes	Yes	No
oracle/wss_username_token_over_ssl_client_template	oracle/wss_username_token_over_ssl_service_template	No	Yes	Yes	No
oracle/wss10_saml_hok_token_with_message_protection_client_template	oracle/wss10_saml_hok_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss10_saml_token_with_message_protection_client_template	oracle/wss10_saml_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss10_saml20_token_with_message_protection_client_template	oracle/wss10_saml20_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss10_username_token_with_message_protection_client_template	oracle/wss10_username_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss10_x509_token_with_message_protection_client_template	oracle/wss10_x509_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss11_kerberos_token_with_message_protection_client_template	oracle/wss11_kerberos_token_with_message_protection_service_template	No	Yes	No	Yes

Table C–41 (Cont.) Message Protection and Authentication Assertion Templates

Client Template	Service Template	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss11_saml_token_with_message_protection_client_template	oracle/wss11_saml_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss11_saml20_token_with_message_protection_client_template	oracle/wss11_saml20_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss11_username_token_with_message_protection_client_template	oracle/wss11_username_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss11_x509_token_with_message_protection_client_template	oracle/wss11_x509_token_with_message_protection_service_template	No	Yes	No	Yes

C.1.3.1 oracle/http_jwt_token_over_ssl_client_template

The http_jwt_token_over_ssl_client_template assertion template includes a JWT token in the HTTP header. The JWT token is created automatically. The issuer name and subject name are provided either programmatically or declarative through the policy. A policy created using this template can be attached to any HTTP-based client. You can specify the audience restriction condition using the configuration override property.

Settings

Table C–42 lists the settings for the http_jwt_token_over_ssl_client_template assertion template.

Table C–42 http_jwt_token_over_ssl_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ■ basic—Client authenticates itself by transmitting the username and password. **Note**: It is recommended that you configure SSL when using basic authentication. For more information, see "Configuring Keystores for SSL" on page 10-36. ■ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ■ custom—**Not supported in this release**. Custom authentication mechanism. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ■ jwt—Client authenticates itself using JWT token. ■ oam—Client authenticates itself using OAM agent. ■ saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. ■ spnego—Client authenticates itself using Kerberos SPNEGO.	`<orasp:auth-header orasp:mechanism="jwt"/>`
Authentication Header—Header Name	Name of the authentication header.	None
Authentication Header—algorithm-suite	Flag that specifies the algorithm suite used to sign the JWT token.	`<orasp:auth-header orasp:algorithm-suite="Basic128Sha256Rsa15"/"` **Note:** This is the only supported value. If any value other than this default value is specified, the policy will fail.
Authentication Header—is-signed	Flag that specifies whether the JWT token is signed. The only valid value for JWT policies is: `true`.	`<orasp:auth-header orasp:is-signed="true"/>`
Authentication Header— is encrypted	Flag that specifies whether the JWT token is encrypted.	`<orasp:auth-header orasp:is-encrypted="false"/>`

Table C–42 (Cont.) http_jwt_token_over_ssl_client_template Settings

Name	Description	Default Value
Transport Security	Flag that specifies whether SSL is enabled.	`<orasp:auth-header orasp:require-tls/>`
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: ■ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ■ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	`<orasp:auth-header orasp:mutual-auth="false"/>`
Transport Security—Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	`<orasp:auth-header orasp:include-timestamp="false"/>`

Configurations

Table C–43 lists the configuration properties and the default settings for the http_jwt_token_over_ssl_client_template assertion template.

Table C–43 http_jwt_token_over_ssl_client_template Configuration Properties

Name	Default Values

audience.uri	Audience restriction. The following conditions are supported:
	- If this property is not set, the service URL is used as the audience URI
	- If this property is set to `NONE` (not case sensitive), then the audience URI is set to null.
	- If this property is set to a value other than `NONE`, then the audience URI is set to this value.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="audience.uri" orawsp:type="string">`
	`  <orawsp:Value/>`
	`</orawsp:Property>`

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="csf.map" orawsp:type="string"/>`
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as a `Value` in this property. For example:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services (OPSS) identity store.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf-key" orawsp:type="string">`
	`   <orawsp:Value>basic.credentials</orawsp:Value>`
	`</orawsp:Property>`

issuer.name	Name of the JWT issuer. The default value is www.oracle.com.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="issuer.name" orawsp:type="string">`
	`  <orawsp:Value>www.oracle.com</orawsp:Value>`
	`</orawsp:Property>`

keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="keystore.sig.csf.key" orawsp:type="string"/>`

Table C–43 (Cont.) http_jwt_token_over_ssl_client_template Configuration Properties

Name	Default Values

propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="propagate.identity.context" orawsp:type="string">`
	`<orawsp:Value/>`

reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope.
	The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1.
	For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="reference.priority" orawsp:type="string"/>`

subject.precedence	Property that specifies the location from which the subject used to create the JWT token should be obtained.
	If `subject.precedence` is set to `true`, the user name to create the JWT token is obtained only from the authenticated Subject. If `subject.precedence` is set to `false`, the user name to create the JWT token is obtained only from the csf-key username property.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="subject.precedence" orawsp:type="string">`
	`  <orawsp:Value>true</orawsp:Value>`
	`</orawsp:Property>`

Table C–43 (Cont.) http_jwt_token_over_ssl_client_template Configuration Properties

Name	Default Values
user.attributes	List of user attributes for the authenticated user to be included in the JWT token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the JWT token. Requires that the Subject is available and `subject.precedence` is set to true. A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create JWT claims. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information. Default setting: `<orawsp:Property orawsp:contentType="optional" orawsp:name="user.attributes" orawsp:type="string"/>`

user.roles.include	User roles to be included in the JWT token. If set to `true`, the authenticated user roles are included in the JWT token as private claims. The default is `false`.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`orawsp:name="user.roles.include" orawsp:type="string">`
	`<orawsp:Value>false</orawsp:Value>`
	`</orawsp:Property>`

user.tenant.name	Reserved for use with Oracle Cloud.

C.1.3.2 oracle/http_jwt_token_over_ssl_service_template

The oracle/http_jwt_token_over_ssl_service_template authenticates users using the username provided in the JWT token in the HTTP header.

Settings

The settings for the http_jwt_token_over_ssl_service_template assertion template are identical to the client version of the assertion template. See Table C–42 for information about the settings.

Configurations

Table C–44 lists the configuration properties and the default settings for the http_jwt_token_over_ssl_service_template assertion template.

Table C–44 http_jwt_token_over_ssl_service_template Configuration Properties

Name	Default Values

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="csf.map" orawsp:type="string"/>`
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as a `Value` in this property. For example:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="keystore.sig.csf.key" orawsp:type="string"/>`

propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	` orawsp:name="propagate.identity.context" orawsp:type="string">`
	`<orawsp:Value/>`

reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope.
	The value of reference.priority can be any number between(-2^{31}) and $(2^{31} - 1)$. The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1.
	For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="reference.priority" orawsp:type="string"/>`

trusted.issuers	A comma-separated list of trusted issuers for an application that will override the trusted issuers defined at the domain level.
	Default setting:
	`<orawsp:Property orawsp:contentType="optional"`
	`  orawsp:name="saml.trusted.issuers" orawsp:type="string">`
	`  <orawsp:Value/>`
	`</orawsp:Property>`

C.1.3.3 oracle/http_oauth2_token_over_ssl_client_template

The http_oauth2_token_over_ssl_client_template assertion template is the HTTP binding level template for OAuth2 token authentication.   This template is same as http_oauth2_token_client_template, except that the AT is propagated over 1-way SSL to the resource.

Settings

Table C–45 lists the settings for the http_oauth2_token_over_ssl_client_template assertion template.

Table C–45 http_oauth2_token_over_ssl_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ▪ basic—Client authenticates itself by transmitting the username and password. **Note**: It is recommended that you configure SSL when using basic authentication. For more information, see "Configuring Keystores for SSL" on page 10-36. ▪ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ▪ custom—**Not supported in this release**. Custom authentication mechanism. ▪ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ▪ jwt—Client authenticates itself using JWT token. ▪ oam—Client authenticates itself using OAM agent. ▪ oauth2—Client authenticates using OAuth2 framework. ▪ saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. ▪ spnego—Client authenticates itself using Kerberos SPNEGO.	`<orasp:auth-header orasp:mechanism="oauth2"/>`
Authentication Header—Header Name	Name of the authentication header.	None
Authentication Header—is-signed	Flag that specifies whether the token is signed.	`<orasp:auth-header orasp:is-signed="false"/>`
Authentication Header— is encrypted	Flag that specifies whether the token is encrypted.	`<orasp:auth-header orasp:is-encrypted="false"/>`

Table C–45 (Cont.) http_oauth2_token_over_ssl_client_template Settings

Name	Description	Default Value
Transport Security	Flag that specifies whether SSL is enabled.	`<orasp:auth-header orasp:require-tls/>`
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: ■ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ■ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	`<orasp:auth-header orasp:mutual-auth="false"/>`
Transport Security—Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	`<orasp:auth-header orasp:include-timestamp="false"/>`

Configurations

The settings for the http_oauth2_token_over_ssl_client_template assertion template are identical to the non-SSL version of the assertion template. See Table C–8 for information about the settings.

C.1.3.4 oracle/wss_http_token_over_ssl_client_template

The wss_http_token_over_ssl_client_template assertion template includes credentials in the HTTP header for outbound client requests and authenticates users against the Oracle Platform Security Services identity store.

Settings

Table C–46 lists the settings for the wss_http_token_over_ssl_client_template assertion template.

Table C–46 wss_http_token_over_ssl_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ■ basic—Client authenticates itself by transmitting the username and password. **Note**: It is recommended that you configure SSL when using basic authentication. For more information, see "Configuring Keystores for SSL" on page 10-36. ■ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ■ custom—**Not supported in this release**. Custom authentication mechanism. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ■ jwt—Client authenticates itself using JWT token. ■ oam—Client authenticates itself using OAM agent. ■ saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. ■ spnego—Client authenticates itself using Kerberos SPNEGO.	basic
Authentication Header—Header Name	Name of the authentication header.	None
Transport Security	Flag that specifies whether SSL is enabled.	Enabled
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: ■ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ■ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	Disabled
Transport Security—Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Disabled

Configurations

Table C–47 lists the configuration properties and the default settings for the wss_http_token_over_ssl_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–47 wss_http_token_over_ssl_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—Not set ■ Default—basic.credentials ■ ContentType—Required ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.3.5 oracle/wss_http_token_over_ssl_service_template

The wss_http_token_over_ssl_service_template assertion template extracts the credentials in the HTTP header and authenticates users against the Oracle Platform Security Services identity store.

Settings

The settings for the wss_http_token_over_ssl_service_template assertion template are identical to the client version of the assertion template. See Table C–46 for information on the settings.

Configurations

Table C–48 lists the configuration properties and the default settings for the wss_http_token_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–48 wss_http_token_over_ssl_service_template Configurations

Name	Description
realm	HTTP Realm. Default settings: ■ Value—Not set ■ Default—owsm ■ ContentType—Constant ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.3.6 oracle/wss_saml_token_bearer_service_template

The wss_saml_token_bearer_service template assertion template authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

Settings

Table C–49 lists the settings for the wss_saml_token_bearer_service_template assertion template.

Table C–49 wss_saml_token_bearer_service template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is: 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: ■ sender-vouches—Uses the Sender Vouches SAML token for authentication.	sender-vouches
Is Signed	Flag that specifies whether the SAML token is signed. The only valid value for this policy is True.	True
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False

Configuration

Table C–50 lists the configuration properties and the default settings for the wss_saml_token_bearer_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–50 wss_saml_token_bearer_service template Configurations

Name	Description
role	SOAP role. Default settings: - Value—Not set - Default—ultimateReceiver - ContentType—Constant - Description—Not set
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: - Value—Not set - Default—null - ContentType—Optional - Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.
user.tenant.name	Reserved for use with Oracle Cloud.

C.1.3.7 oracle/wss_saml_token_bearer_over_ssl_client_template

The wss_saml_token_bearer_over_ssl_client template assertion template includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method [*Bearer*] is created automatically.

Settings

Table C–51 lists the settings for the wss_saml_token_bearer_over_ssl_client_template assertion template.

Table C–51 wss_saml_token_bearer_over_ssl_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is: 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: bearer.	bearer
Is Signed	Flag that specifies whether the SAML token is signed.	False
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is true, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName	unspecified
Transport Security	Flag that specifies whether SSL is enabled.	Enabled
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: ■ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ■ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	Disabled
Transport Security—Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Disabled

Configurations

Table C–52 lists the configuration properties and the default settings for the wss_saml_token_bearer_over_ssl_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–52 wss_saml_token_bearer_over_ssl_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to true. Default settings: ■ Value—Not set ■ Default—Null. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
saml.issuer.name	Issuer URI. Default settings: ■ Value—Not set ■ Default—www.oracle.com ■ ContentType—Optional ■ Description—Not set
user.roles.include	User roles to be included. Default settings: ■ Value—Not set ■ Default—false ■ ContentType—Optional ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–52 (Cont.) wss_saml_token_bearer_over_ssl_client_template Configurations

Name	Description
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is true, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—true ■ ContentType—Optional ■ Description—Not set
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—null ■ ContentType—Optional ■ Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–52 (Cont.) wss_saml_token_bearer_over_ssl_client_template Configurations

Name	Description
saml.envelope.signature.required	Flag that specifies whether the bearer token is signed using the domain signature key. You can override the domain signature key using the private signature key configured using `keystore.sig.csf.key`. Set this flag `false` (in both client and service policy) to have the bearer token be unsigned. Default setting: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.
user.tenant.name	Reserved for use with Oracle Cloud.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.8 oracle/wss_saml_token_bearer_over_ssl_service_template

The wss_saml_token_bearer_over_ssl_service_template assertion template authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

Settings

The settings for the wss_saml_token_bearer_over_ssl_service_template assertion template are identical to the client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–51 for information on the settings.

Configurations

Table C–53 lists the configuration properties and the default settings for the wss_saml_token_bearer_over_ssl_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–53 wss_saml_token_bearer_over_ssl_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.3.9 oracle/wss_saml20_token_bearer_over_ssl_client_template

The wss_saml20_token_bearer_over_ssl_client template assertion template includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method [*Bearer*] is created automatically.

Settings

Table C–54 lists the settings for the wss_saml20_token_bearer_over_ssl_client_template assertion template.

Table C–54 wss_saml20_token_bearer_over_ssl_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is: 2.0.	2.0
Confirmation Type	Confirmation type. The only valid value is: bearer.	bearer
Is Signed	Flag that specifies whether the SAML token is signed.	False
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the `csf-key` property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName ■ kerberos	unspecified
Transport Security	Flag that specifies whether SSL is enabled.	Enabled
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: ■ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ■ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	Disabled
Transport Security—Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Disabled

Configurations

Table C–55 lists the configuration properties and the default settings for the wss_saml20_token_bearer_over_ssl_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–55 wss_saml20_token_bearer_over_ssl_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to true. Default settings: ■ Value—Not set ■ Default—Not set. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
saml.issuer.name	Issuer URI. Default settings: ■ Value—Not set ■ Default—www.oracle.com ■ ContentType—Optional ■ Description—Not set
user.roles.include	User roles to be included. Default settings: ■ Value—Not set ■ Default—false ■ ContentType—Optional ■ Description—Not set

Table C–55 (Cont.) wss_saml20_token_bearer_over_ssl_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is true, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—null ■ ContentType—Optional ■ Description—Not set

Table C–55 (Cont.) wss_saml20_token_bearer_over_ssl_client_template Configurations

Name	Description
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
saml.envelope.signature.required	Flag that specifies whether the bearer token is signed using the domain signature key. You can override the domain signature key using the private signature key configured using `keystore.sig.csf.key`. Set this flag `false` (in both client and service policy) to have the bearer token be unsigned. Default setting: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.
user.tenant.name	Reserved for use with Oracle Cloud.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.10 oracle/wss_saml20_token_bearer_over_ssl_service_template

The wss_saml20_token_bearer_over_ssl_service_template assertion template authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

Settings

The settings for the wss_saml20_token_bearer_over_ssl_service_template assertion template are identical to the client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–54 for information on the settings.

Configurations

Table C–56 lists the configuration properties and the default settings for the wss_saml20_token_bearer_over_ssl_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–56 wss_saml20_token_bearer_over_ssl_service_template Configurations

Name	Description
role	SOAP role. Default settings: - Value—Not set - Default—ultimateReceiver - ContentType—Constant - Description—Not set
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: - Value—Not set - Default—null - ContentType—Optional - Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.3.11 oracle/wss_saml_token_over_ssl_client_template

The wss_saml_token_over_ssl_client_template assertion template enables the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type.

Settings

Table C–57 lists the settings for the wss_saml_token_over_ssl_client_template assertion template.

Table C–57 wss_saml_token_over_ssl_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is: 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: ▪ sender-vouches—Uses the Sender Vouches SAML token for authentication.	sender-vouches
Is Signed	Flag that specifies whether the SAML token is signed. The only valid value for this policy is True.	True
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is true, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ▪ unspecified ▪ emailAddress ▪ X509SubjectName ▪ WindowsDomainQualifiedName	unspecified
Transport Security	Flag that specifies whether SSL is enabled.	Enabled
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: ▪ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ▪ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	Enabled
Transport Security—Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Disabled

Configurations

Table C–58 lists the configuration properties and the default settings for the wss_saml_token_over_ssl_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–58 wss_saml_token_over_ssl_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to true. Default settings: - Value—Not set - Default—Not set. Attribute names should be comma separated. - ContentType—Optional - Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
saml.issuer.name	Issuer URI. Default settings: - Value—Not set - Default—www.oracle.com - ContentType—Optional - Description—Not set
user.roles.include	User roles to be included. Default settings: - Value—Not set - Default—false - ContentType—Optional - Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–58 (Cont.) wss_saml_token_over_ssl_client_template Configurations

Name	Description
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is true, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

Table C–58 (Cont.) wss_saml_token_over_ssl_client_template Configurations

Name	Description
user.tenant.name	Reserved for use with Oracle Cloud.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.12 oracle/wss_saml_token_over_ssl_service_template

The wss_saml_token_over_ssl_service_template enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type.

Settings

The settings for the wss_saml_token_over_ssl_service_template assertion template are identical to the client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–57 for information on the settings.

Configurations

Table C–59 lists the configuration properties and the default settings for the wss_saml_token_over_ssl_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–59 wss_saml_token_over_ssl_service_template Configurations

Name	Description
role	SOAP role. Default settings: ▪ Value—Not set ▪ Default—ultimateReceiver ▪ ContentType—Constant ▪ Description—Not set
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: ▪ Value—Not set ▪ Default—null ▪ ContentType—Optional ▪ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.3.13 oracle/wss_saml20_token_over_ssl_client_template

The wss_saml20_token_over_ssl_client_template assertion template enables the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type.

Settings

Table C–60 lists the settings for the wss_saml20_token_over_ssl_client_template assertion template.

Table C–60 wss_saml20_token_over_ssl_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is: 2.0.	2.0
Confirmation Type	Confirmation type. The only valid value is: ■ sender-vouches—Uses the Sender Vouches SAML token for authentication.	sender-vouches
Is Signed	Flag that specifies whether the SAML token is signed. The only valid value for this policy is True.	True
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is true, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName ■ kerberos	unspecified
Transport Security	Flag that specifies whether SSL is enabled.	Enabled
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: ■ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ■ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	Enabled
Transport Security—Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Disabled

Configurations

Table C–61 lists the configuration properties and the default settings for the wss_saml20_token_over_ssl_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–61 wss_saml20_token_over_ssl_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to true. Default settings: ■ Value—Not set ■ Default—Not set. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
saml.issuer.name	Issuer URI. Default settings: ■ Value—Not set ■ Default—www.oracle.com ■ ContentType—Optional ■ Description—Not set
user.roles.include	User roles to be included. Default settings: ■ Value—Not set ■ Default—false ■ ContentType—Optional ■ Description—Not set

Table C–61 (Cont.) wss_saml20_token_over_ssl_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—Not set ■ Default—true ■ ContentType—Optional ■ Description—Not set
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

Table C–61 (Cont.) wss_saml20_token_over_ssl_client_template Configurations

Name	Description
user.tenant.name	Reserved for use with Oracle Cloud.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.14 oracle/wss_saml20_token_over_ssl_service_template

The wss_saml20_token_over_ssl_service_template enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type.

Settings

The settings for the wss_saml20_token_over_ssl_service_template assertion template are identical to the client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–60 for information on the settings.

Configurations

Table C–62 lists the configuration properties and the default settings for the wss_saml20_token_over_ssl_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–62 wss_saml20_token_over_ssl_service_template Configurations

Name	Description
role	SOAP role. Default settings: - Value—Not set - Default—ultimateReceiver - ContentType—Constant - Description—Not set
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.3.15 oracle/wss_username_token_over_ssl_client_template

The wss_username_token_over_ssl_client_template assertion template includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages. The assertion supports three types of password credentials: plain text, digest, and no password.

> **Note:** Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token.

Settings

Table C–63 lists the settings for the wss_username_token_over_ssl_client_template assertion template.

Table C–63 wss_username_token_over_ssl_client_template Settings

Name	Description	Default Value
Password Type	Type of password required. Valid values are: - none—No password. - plaintext—Password in clear text. - digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. **Note**: The plaintext type is not recommended when the token propagation occurs on an unsecure channel. However, if SSL is being used as the transport channel to secure a point-to-point connection between client and server, the plaintext type can be used as the channel takes care of protecting the password.	plaintext
Creation Time Required	Flag that specifies whether a time stamp for the creation of the username token is required. **Notes**: - If Password Type is set to digest, then this attribute must be set to `true`. Otherwise, the policy to which it is attached will not validate. - If Nonce Required is set to `true`, than this attribute must be set to `true`. Otherwise, nonce will be cached forever to prevent replay attacks.	False
Nonce Required	Flag that specifies whether a nonce must be included with the username to prevent replay attacks. **Notes**: - If Password Type is set to digest, then this attribute must be set to `true`. Otherwise, the policy to which it is attached will not validate. - If Creation Time Required is set to `true`, than this attribute must be set to `true`. Otherwise, nonce will be cached forever to prevent replay attacks.	False
Transport Security	Flag that specifies whether SSL is enabled.	Enabled
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: - Enabled—Two-way authentication. The service must authenticate itself to the client, and the client must authenticate itself to the service. - Disabled—One-way authentication. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	Disabled
Transport Security—Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Disabled

Configurations

Table C–64 lists the configuration properties and the default settings for the wss_username_token_over_ssl_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–64 wss_username_token_over_ssl_client_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services (OPSS) identity store. Default settings: ■ Value—Not set ■ Default—basic.credentials ■ ContentType—Required ■ Description—Not set

Table C–64 (Cont.) wss_username_token_over_ssl_client_template Configurations

Name	Description
user.tenant.name	Reserved for use with Oracle Cloud.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.16 oracle/wss_username_token_over_ssl_service_template

The wss_username_token_over_ssl_service_template assertion template uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users against the Oracle Platform Security Services configured identity store. The assertion supports three types of password credentials: plain text, digest, and no password.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token.

Settings

The settings for the wss_username_token_over_ssl_service_template assertion template are identical to the client version of the assertion template. See Table C–63 for information on the settings.

Configurations

Table C–65 lists the configuration properties and the default settings for the wss_username_token_over_ssl_service_template assertion template. For details about the

configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–65 wss_username_token_over_ssl_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.3.17 oracle/wss10_saml_hok_token_with_message_protection_client_template

The wss10_saml_hok_token_with_message_protection_client_template assertion template provides message protection (integrity and confidentiality) and SAML holder of key based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

Settings

Table C–66 lists the settings for the wss10_saml_hok_token_with_message_protection_client_template assertion template.

Table C–66 wss10_saml_hok_token_with_message_protection_client_template Settings

Name	Description	Default Value
SAML Token Type		
Version	SAML version. The only valid value is: 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: holder-of-key.	holder-of-key
Is Signed	Flag that specifies whether the SAML token is signed. The only valid value is: `True`.	True

Table C–66 (Cont.) wss10_saml_hok_token_with_message_protection_client_template Settings

Name	Description	Default Value
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False
Name Identifier Format	Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specifies the type of format to be used for the name identifier. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName	unspecified
X509 Token		
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	ski
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct

Table C–66 (Cont.) wss10_saml_hok_token_with_message_protection_client_template Settings

Name	Description	Default Value
Recipient Sign Key Reference Mechanism	Mechanism used when signing the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–67 lists the configuration properties and the default settings for the wss10_saml_hok_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–67 wss10_saml_hok_token_with_message_protection_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to `true`. Default settings: ■ Value—Not set ■ Default—Not set. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—orakey ■ Default—Not set ■ ContentType—Required ■ Description—Not set
saml.issuer.name	Issuer URI. Default settings: ■ Value—www.oracle.com ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
user.roles.include	User roles to be included. Default settings: ■ Value—false ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–67 (Cont.) wss10_saml_hok_token_with_message_protection_client_template Configurations

Name	Description
saml.assertion.filename	Name of the of the SAML token file. Default settings: ■ Value—temp ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–67 (Cont.) wss10_saml_hok_token_with_message_protection_client_template Configurations

Name	Description
user.tenant.name	Reserved for use with Oracle Cloud.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.18 oracle/wss10_saml_hok_token_with_message_protection_service_template

The wss10_saml_hok_token_with_message_protection_service_template assertion template enforces message-level protection and SAML holder of key based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

Settings

The settings for the wss10_saml_hok_token_with_message_protection_service_template are identical to those for the client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–66 for information on the settings.

Configurations

Table C–68 lists the configuration properties and the default settings for the wss10_saml_hok_token_with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–68 wss10_saml_hok_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–68 (Cont.) wss10_saml_hok_token_with_message_protection_service_template Configurations

Name	Description
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.3.19 oracle/wss10_saml_token_with_message_protection_client_template

The wss10_saml_token_with_message_protection_client_template assertion template provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

The Web service consumer includes a SAML token in the SOAP header, and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

Settings

Table C–69 lists the settings for the wss10_saml_token_with_message_protection_client_template assertion template.

Table C–69 wss10_saml_token_with_message_protection_client_template Settings

Name	Description	Default Value
SAML Token Type		
Version	SAML version. The only valid value is: 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: sender-vouches.	sender-vouches
Is Signed	Flag that specifies whether the SAML token is signed. The only valid value for this policy is: `True`.	True
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName	unspecified
X509 Token		
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when signing the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct

Table C–69 (Cont.) wss10_saml_token_with_message_protection_client_template Settings

Name	Description	Default Value
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–70 lists the configuration properties and the default settings for the wss10_saml_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–70 wss10_saml_token_with_message_protection_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to `true`. Default settings: ▪ Value—Not set ▪ Default—Not set. Attribute names should be comma separated. ▪ ContentType—Optional ▪ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ▪ Value—Not set ▪ Default—orakey ▪ ContentType—Required ▪ Description—Not set
saml.issuer.name	Issuer URI. Default settings: ▪ Value—Not set ▪ Default—www.oracle.com ▪ ContentType—Optional ▪ Description—Not set
user.roles.include	User roles to be included. Default settings: ▪ Value—Not set ▪ Default—false ▪ ContentType—Optional ▪ Description—Not set

Table C–70 (Cont.) wss10_saml_token_with_message_protection_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–70 (Cont.) wss10_saml_token_with_message_protection_client_template Configurations

Name	Description
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—null ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

Table C–70 (Cont.) wss10_saml_token_with_message_protection_client_template Configurations

Name	Description
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.
user.tenant.name	Reserved for use with Oracle Cloud.

C.1.3.20 oracle/wss10_saml_token_with_message_protection_service_template

The wss10_saml_token_with_message_protection_service_template assertion template enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

The Web service consumer includes a SAML token in the SOAP header, and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

Settings

The settings for the wss10_saml_token_with_message_protection_service_template are identical to those for client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–69 for information on the settings.

Configurations

Table C–71 lists the configuration properties and the default settings for the wss10_saml_token_with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–71 wss10_saml_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–71 (Cont.) wss10_saml_token_with_message_protection_service_template Configurations

Name	Description
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.3.21 oracle/wss10_saml20_token_with_message_protection_client_template

The wss10_saml20_token_with_message_protection_client_template assertion template provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

The Web service consumer includes a SAML token in the SOAP header, and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

Settings

Table C–72 lists the settings for the wss10_saml20_token_with_message_protection_client_template assertion template.

Table C–72 wss10_saml20_token_with_message_protection_client_template Settings

Name	Description	Default Value
SAML Token Type		
Version	SAML version. The only valid value is: 2.0.	2.0
Confirmation Type	Confirmation type. The only valid value is: sender-vouches.	sender-vouches
Is Signed	Flag that specifies whether the SAML token is signed. The only valid value for this policy is: `True`.	True
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName ■ kerberos	unspecified
X509 Token		
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when signing the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct

Table C–72 (Cont.) wss10_saml20_token_with_message_protection_client_template Settings

Name	Description	Default Value
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–73 lists the configuration properties and the default settings for the wss10_saml20_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–73 wss10_saml20_token_with_message_protection_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to `true`. Default settings: ■ Value—Not set ■ Default—Not set. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—Not set ■ Default—orakey ■ ContentType—Required ■ Description—Not set
user.roles.include	User roles to be included. Default settings: ■ Value—Not set ■ Default—false ■ ContentType—Optional ■ Description—Not set
saml.issuer.name	Issuer URI. Default settings: ■ Value—Not set ■ Default—www.oracle.com ■ ContentType—Optional ■ Description—Not set

Table C–73 (Cont.) wss10_saml20_token_with_message_protection_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: - Value—basic.credentials - Default—Not set - ContentType—Optional - Description—Not set

Table C–73 (Cont.) wss10_saml20_token_with_message_protection_client_template Configurations

Name	Description
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
attesting.mapping.attribute	The mapping attribute used to represent the attesting entity. Only the DN is currently supported. This attribute is applicable only to sender vouches and then only to message protection use cases. It is not applicable to SAML over SSL policies. Default settings: ■ Value—DN ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

Table C–73 (Cont.) wss10_saml20_token_with_message_protection_client_template Configurations

Name	Description
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.
user.tenant.name	Reserved for use with Oracle Cloud.

C.1.3.22 oracle/wss10_saml20_token_with_message_protection_service_template

The wss10_saml20_token_with_message_protection_service_template assertion template enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

The Web service consumer includes a SAML token in the SOAP header, and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

Settings

The settings for the wss10_saml20_token_with_message_protection_service_template are similar to those of the client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–72 for information on the settings.

Configurations

Table C–74 lists the configuration properties and the default settings for the wss10_saml20_token_with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–74 wss10_saml20_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Default settings: - Value—Not set - Default—ultimateReceiver - ContentType—Constant - Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set

Table C–74 (Cont.) wss10_saml20_token_with_message_protection_service_template Configurations

Name	Description
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.3.23 oracle/wss10_username_token_with_message_protection_client_template

The wss10_username_token_with_message_protection_client_template assertion template provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Credentials are included in the WS-Security UsernameToken header in the outbound SOAP message.

The assertion supports three types of password credentials: plain text, digest, and no password.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

Settings

Table C–75 lists the settings for the wss10_username_token_with_message_protection_client_template assertion template.

Table C–75 wss10_username_token_with_message_protection_client_template Settings

Name	Description	Default Value
Username Token		
Password Type	Type of password required. Valid values are: ■ none—No password. ■ plaintext—Password in clear text. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest.	plaintext
Creation Time Required	Flag that specifies whether a time stamp for the creation of the username token is required. **Notes**: ■ If Password Type is set to digest, then this attribute must be set to `true`. Otherwise, the policy to which it is attached will not validate. ■ If Nonce Required is set to `true`, than this attribute must be set to `true`. Otherwise, nonce will be cached forever to prevent replay attacks.	False
Nonce Required	Flag that specifies whether a nonce must be included with the username to prevent replay attacks. **Notes**: ■ If Password Type is set to digest, then this attribute must be set to `true`. Otherwise, the policy to which it is attached will not validate. ■ If Creation Time Required is set to `true`, than this attribute must be set to `true`. Otherwise, nonce will be cached forever to prevent replay attacks.	False
Is Signed	Flag that specifies whether the username is signed.	True
Is Encrypted	Flag that specifies whether the username is encrypted.	True
X509 Token		
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when signing the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct

Table C–75 (Cont.) wss10_username_token_with_message_protection_client_template Settings

Name	Description	Default Value
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–76 lists the configuration properties and the default settings for the wss10_username_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–76 wss10_username_token_with_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—Not set ■ Default—orakey ■ ContentType—Required ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—Not set ■ Default—basic.credentials ■ ContentType—Required ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–76 (Cont.) wss10_username_token_with_message_protection_client_template Configurations

Name	Description
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.24 oracle/wss10_username_token_with_message_protection_service_template

The wss10_username_token_with_message_protection_service_template assertion template enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

The assertion supports three types of password credentials: plain text, digest, and no password.

> **Note:** Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The

Web service provider decrypts the message, and verifies and authenticates the signature.

Settings

The settings for the wss10_username_token_with_message_protection_service_template assertion template are identical to the client version of the assertion template. See Table C–75 for information on the settings.

Configurations

Table C–77 lists the configuration properties and the default settings for the wss10_username_token_with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–77 wss10_username_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–77 (Cont.) wss10_username_token_with_message_protection_service_template Configurations

Name	Description
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.3.25 oracle/wss10_x509_token_with_message_protection_client_template

The wss10_x509_token_with_message_protection_client template assertion template provides message protection (integrity and confidentiality) and certificate credential population for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

Settings

Table C–78 lists the settings for the wss10_x509_token_with_message_protection_client template assertion template.

Table C–78 wss10_x509_token_with_message_protection_client_template Settings

Name	Description	Default Value
X509 Token		
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when signing the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–79 lists the configuration properties and the default settings for the wss10_x509_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–79 wss10_x509_token_with_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—Not set ■ Default—orakey ■ ContentType—Required ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–79 (Cont.) wss10_x509_token_with_message_protection_client_template Configurations

Name	Description
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.26 oracle/wss10_x509_token_with_message_protection_service_template

The wss10_x509_token_with_message_protection_service_template assertion template enforces message protection (integrity and confidentiality) and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

Settings

The settings for the wss10_x509_token_with_message_protection_service_template assertion template are identical to the client version of the assertion template. See Table C–78 for information on the settings.

Configurations

Table C–80 lists the configuration properties and the default settings for the wss10_x509_token_with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–80 wss10_x509_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

Table C–80 (Cont.) wss10_x509_token_with_message_protection_service_template Configurations

Name	Description
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.3.27 oracle/wss11_kerberos_token_with_message_protection_client_template

The wss11_kerberos_token_with_message_protection_client_template assertion template includes a Kerberos token in the WS-Security header in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

Settings

Table C–81 lists the settings for the wss11_kerberos_token_with_message_protection_client_template assertion template.

Table C–81 wss11_kerberos_token_with_message_protection_client_template Settings

Name	Description	Default Value
Kerberos Token Type	Type of Kerberos token. The only valid value is: gss-apreq-v5 (Kerberos Version 5 GSS-API).	gss-apreq-v5
X509 Token		
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	TripleDes
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	Enabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–82 lists the configuration properties and the default settings for the wss11_kerberos_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–82 wss11_kerberos_token_with_message_protection_client_template Configurations

Name	Description
service.principal.name	Kerberos principal name that identifies the service. Default settings: ▪ Value—HOST/localhost@EXAMPLE.COM ▪ Default—Not set ▪ ContentType—Required ▪ Description—Not set
keytab.location	Location of the client's keytab file. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set

Table C–82 (Cont.) wss11_kerberos_token_with_message_protection_client_template Configurations

Name	Description
caller.principal.name	Client's principal name as generated using the `ktpass` command and mapped to the username for which the kerberos token should be generated. Use the following format: `<username>@<REALM NAME>`. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set **Note:** `keytab.location` and `caller.principal.name` are required for propagating client identity for Java EE applications.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.28 oracle/wss11_kerberos_token_with_message_protection_service_template

The wss11_kerberos_token_with_message_protection_service_template assertion template enforces in accordance with the WS-Security Kerberos Token Profile v1.1 standard. It extracts the Kerberos token from the SOAP header and authenticates the user. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services.

Settings

The settings for the wss11_keberos_token_with_message_protection_service_template are identical to the client version of the assertion template. See Table C–81 for information on the settings.

Configurations

None required.

C.1.3.29 oracle/wss11_saml_token_with_message_protection_client_template

The wss11_saml_token_with_message_protection_client_template assertion template enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests in accordance with WS-Security 1.1. A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation.

> **Note:** This template is also the basis for the wss11_saml_token_with_message_protection_identity_switch_client_policy, which can perform dynamic identity switching by propagating a different identity than the one based on authenticated Subject. However, in the assertion content the `subject.precedence` config-override property defaults to `false`.

Settings

Table C–83 lists the settings for the wss11_saml_token_with_message_protection_client_template assertion template.

Table C–83 wss11_saml_token_with_message_protection_client_template Settings

Name	Description	Default Value
SAML Token Type		
Version	SAML version. The only valid value is: 1.1.	None
Confirmation Type	Confirmation type. Valid values include: sender-vouches.	sender-vouches.
Is Signed	Flag that specifies whether the SAML token is signed. The only valid value for SAML policies is: `True`.	True
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName	unspecified
X509 Token		

Table C–83 (Cont.) wss11_saml_token_with_message_protection_client_template Settings

Name	Description	Default Value
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead. This value is valid for Encryption Key Reference Mechanism only (described below.)	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	thumbprint
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	Enabled
Derived Keys	Flag that specifies whether derived keys should be used.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–84 lists the configuration properties and the default settings for the wss11_saml_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–84 wss11_saml_token_with_message_protection_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to `true`. Default settings: ■ Value—Not set ■ Default—Not set. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
saml.issuer.name	Issuer URI. Default settings: ■ Value—Not set ■ Default—www.oracle.com ■ ContentType—Optional ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—Not set ■ Default—orakey ■ ContentType—Required ■ Description—Not set

Table C–84 (Cont.) wss11_saml_token_with_message_protection_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–84 (Cont.) wss11_saml_token_with_message_protection_client_template Configurations

Name	Description
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set **Note:** When configuring a wss11_saml_token_identity_switch_with_mesage_protection_client_policy, the `subject.precedence` value defaults to `false` for dynamic identity switching.
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

Table C–84 (Cont.) wss11_saml_token_with_message_protection_client_template Configurations

Name	Description
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.
user.tenant.name	Reserved for use with Oracle Cloud.

C.1.3.30 oracle/wss11_saml_token_with_message_protection_service_template

The wss11_saml_token_with_message_protection_service_template assertion template enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

Settings

The settings for the wss11_saml_token_with_message_protection_service_template are identical to the client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–83 for information on the settings.

Configurations

Table C–85 lists the configuration properties and the default settings for the wss11_saml_token__with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–85 wss11_saml_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Default settings: - Value—Not set - Default—ultimateReceiver - ContentType—Constant - Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set

Table C–85 (Cont.) wss11_saml_token_with_message_protection_service_template Configurations

Name	Description
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.3.31 oracle/wss11_saml20_token_with_message_protection_client_template

The wss11_saml20_token_with_message_protection_client_template assertion template enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests in accordance with WS-Security 1.1. A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation.

Settings

Table C–86 lists the settings for the wss11_saml20_token_with_message_protection_client_template assertion template.

Table C–86 wss11_saml20_token_with_message_protection_client_template Settings

Name	Description	Default Value
SAML Token Type		
Version	SAML version. The only valid value is: 2.0.	2.0
Confirmation Type	Confirmation type. Valid values include: sender-vouches.	sender-vouches.
Is Signed	Flag that specifies whether the SAML token is signed. The only valid value for SAML policies is: `True`.	True
Is Encrypted	Flag that specifies whether the SAML token is encrypted.	False

Table C–86 (Cont.) wss11_saml20_token_with_message_protection_client_template Settings

Name	Description	Default Value
Name Identifier Format	Specifies the type of format to be used for the name identifier. Name Identifier Format is applicable only when `subject.precedence` is set to `false`. If `subject.precedence` is `false`, the user name to create the SAML assertion is obtained from the csf-key property or the username property (see "Configure the Username for the SAML Assertion" on page 10-65). The format of the user name must be the same as the format set in Name Identifier Format. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained from the Subject. In this case, the Name Identifier Format is always "unspecified" and this cannot be changed by setting Name Identifier Format. Specify one of the following values: ■ unspecified ■ emailAddress ■ X509SubjectName ■ WindowsDomainQualifiedName ■ kerberos	unspecified
X509 Token		
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead. This value is valid for Encryption Key Reference Mechanism only (described below.)	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	thumbprint
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled

Table C–86 (Cont.) wss11_saml20_token_with_message_protection_client_template Settings

Name	Description	Default Value
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	Enabled
Derived Keys	Flag that specifies whether derived keys should be used.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–87 lists the configuration properties and the default settings for the wss11_saml20_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–87 wss11_saml20_token_with_message_protection_client_template Configurations

Name	Description
user.attributes	User attributes related to the principal of the SAML token. Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion. Requires that the Subject is available and `subject.precedence` is set to `true`. Default settings: ■ Value—Not set ■ Default—Not set. Attribute names should be comma separated. ■ ContentType—Optional ■ Description—Not set A client policy reads the values of the attributes specified using `user.attributes` from the configured identity store. All valid attribute names and values are used to create the SAML attribute statement. The `user.attributes` property is supported for a single identity store, and only the first identity store in the list is used. The user must therefore exist and be valid in the identity store used by the configured WebLogic Server Authentication provider. Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 10-59. If the identity store you require is not the first identity store, you can specify that additional identity stores be searched. See "Including User Attributes in the Assertion" on page 10-66 for more information.
saml.issuer.name	Issuer URI. Default settings: ■ Value—Not set ■ Default—www.oracle.com ■ ContentType—Optional ■ Description—Not set
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—Not set ■ Default—orakey ■ ContentType—Required ■ Description—Not set

Table C–87 (Cont.) wss11_saml20_token_with_message_protection_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—basic.credentials ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–87 (Cont.) wss11_saml20_token_with_message_protection_client_template Configurations

Name	Description
subject.precedence	Set `subject.precedence` to `false` to allow for the use of a client-specified username rather than the authenticated subject. If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained only from the Subject. Similarly, if `subject.precedence` is `false`, the user name to create the SAML assertion is obtained only from the csf-key username property. Default settings: ■ Value—true ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
attesting.mapping.attribute	The mapping attribute used to represent the attesting entity. Only the DN is currently supported. This attribute is applicable only to sender vouches and then only to message protection use cases. It is not applicable to SAML over SSL policies. Default settings: ■ Value—DN ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

Table C–87 (Cont.) wss11_saml20_token_with_message_protection_client_template Configurations

Name	Description
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.
user.tenant.name	Reserved for use with Oracle Cloud.

C.1.3.32 oracle/wss11_saml20_token_with_message_protection_service_template

The wss11_saml20_token_with_message_protection_service_template assertion template enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

Settings

The settings for the wss11_saml_token_with_message_protection_service_template are similar to the client version of the assertion template, with the exception that Name Identifier Format is not present. See Table C–85 for information on the settings.

Configurations

Table C–88 lists the configuration properties and the default settings for the wss11_saml20_token__with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–88 wss11_saml20_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–88 (Cont.) wss11_saml20_token_with_message_protection_service_template Configurations

Name	Description
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes. Default is `false`.

C.1.3.33 oracle/wss11_username_token_with_message_protection_client_template

The ws11_username_token_with_message_protection_client_template assertion template includes authentication and message protection in accordance with the WS-Security v1.1 standard.

The Web service consumer inserts username and password credentials, and signs and encrypts the outgoing SOAP message. The Web service provider decrypts and verifies the message and the signature.

To prevent replay attacks, the assertion provides the option to include time stamps and verification by the Web service provider. The message can be protected with ciphers of different strengths.

Settings

Table C–89 lists the settings for the wss11_username_token_with_message_protection_client_template assertion template.

Table C–89 wss11_username_token_with_message_protection_client_template Settings

Name	Description	Default Value
Username Token		
Password Type	Type of password required. Valid values are: ■ none—No password. ■ plaintext—Password in clear text. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest.	plaintext
Creation Time Required	Flag that specifies whether a time stamp for the creation of the username token is required. **Notes**: ■ If Password Type is set to digest, then this attribute must be set to `true`. Otherwise, the policy to which it is attached will not validate. ■ If Nonce Required is set to `true`, than this attribute must be set to `true`. Otherwise, nonce will be cached forever to prevent replay attacks.	False
Nonce Required	Flag that specifies whether a nonce must be included with the username to prevent replay attacks. **Notes**: ■ If Password Type is set to digest, then this attribute must be set to `true`. Otherwise, the policy to which it is attached will not validate. ■ If Creation Time Required is set to `true`, than this attribute must be set to `true`. Otherwise, nonce will be cached forever to prevent replay attacks.	False
Is Signed	Flag that specifies whether the username is signed.	True
Is Encrypted	Flag that specifies whether the username is encrypted.	True
X509 Token		
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead.	thumbprint
Message Security		

Table C–89 (Cont.) wss11_username_token_with_message_protection_client_template Settings

Name	Description	Default Value
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	Enabled
Derived Keys	Flag that specifies whether derived keys should be used.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–90 lists the configuration properties and the default settings for the wss11_username_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–90 wss11_username_token_with_message_protection_client_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Default settings: ■ Value—Not set ■ Default—basic.credentials ■ ContentType—Required ■ Description—Not set
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—Not set ■ Default—orakey ■ ContentType—Required ■ Description—Not set
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–90 (Cont.) wss11_username_token_with_message_protection_client_template Configurations

Name	Description
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.
user.tenant.name	Reserved for use with Oracle Cloud.

C.1.3.34 oracle/wss11_username_token_with_message_protection_service_template

The ws11_username_token_with_message_protection_service_template assertion template enforces authentication and message protection in accordance with the WS-Security v1.1 standard.

The Web service consumer inserts username and password credentials, and signs and encrypts the outgoing SOAP message. The Web service provider decrypts and verifies the message and the signature. To prevent replay attacks, the assertion provides the option to include time stamps and verification by the Web service provider. The message can be protected with ciphers of different strengths.

Settings

The settings for the wss11_username_token_with_message_protection_service_template are identical to the client version of the assertion template. See Table C–89 for information on the settings.

Configurations

Table C–91 lists the configuration properties and the default settings for the wss11_username_token_with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–91 wss11_username_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set

Table C–91 (Cont.) wss11_username_token_with_message_protection_service_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.3.35 oracle/wss11_x509_token_with_message_protection_client_template

The wss11_x509_token_with_message_protection_client_template assertion template provides message protection (integrity and confidentiality) and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard. Credentials are included in the WS-Security binary security token of the SOAP message.]

Settings

Table C–92 lists the settings for the wss11_x509_token_with_message_protection_client_template assertion template.

Table C–92 wss11_x509_token_with_message_protection_client_template Settings

Name	Description	Default Value
X509 Token		
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ▪ direct—X.509 Token is included in the request. ▪ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ▪ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ▪ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead. This value is valid for Encryption Key Reference Mechanism only (described below.)	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	thumbprint
Message Security		
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Encrypt Signature	Flag that specifies whether to encrypt the signature.	Disabled
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	Enabled
Derived Keys	Flag that specifies whether derived keys should be used.	Disabled
Request Message Settings	See Table C–118.	N/A
Response Message Settings	See Table C–118.	N/A
Fault Message Settings	See Table C–118.	N/A

Configurations

Table C–93 lists the configuration properties and the default settings for the wss11_x509_token_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–93 wss11_x509_token_with_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ▪ Value—Not set ▪ Default—orakey ▪ ContentType—Required ▪ Description—Not set
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ▪ Value—Not set ▪ Default—Not set ▪ ContentType—Optional ▪ Description—Not set

Table C–93 (Cont.) wss11_x509_token_with_message_protection_client_template Configurations

Name	Description
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this csf-key is fetched from the keystore and used for signing. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.3.36 oracle/wss11_x509_token_with_message_protection_service_template

The wss11_x509_token_with_message_protection_service_template assertion template enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. The certificate is extracted from the WS-Security binary security token header, and the credentials in the certificate are validated against the Oracle Platform Security Services identity store.

Settings

The settings for the wss11_x509_token_with_message_protection_service_template are identical to the client version of the assertion template. See Table C–92 for information on the settings.

Configurations

Table C–94 lists the configuration properties and the default settings for the wss11_x509_token_with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–94 wss11_x509_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Default settings: ▪ Value—Not set ▪ Default—ultimateReceiver ▪ ContentType—Constant ▪ Description—Not set

Table C–94 (Cont.) wss11_x509_token_with_message_protection_service_template Configurations

Name	Description
csf.map	Oracle WSM map in the credential store that contains the CSF aliases. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`. Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.
keystore.enc.csf.key	The alias and password used for storing the decryption key password in the keystore. If you set this value you then can override it, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: - Value—Not set - Default—Not set - ContentType—Optional - Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.4 WS-Trust Assertion Templates

Table C–95 summarizes the WS-Trust assertion templates.

In this release, you can use Fusion Middleware Control to directly edit the assertion template text, but the Settings and Configurations pages are not available.

Table C–95 WS-Trust Assertion Templates

Name	Description
oracle/sts_trust_config_client_template	STS configuration information assertion template that is used to invoke STS for token exchange.
oracle/sts_trust_config_service_template	STS configuration information assertion template that is used to invoke STS for token exchange.
oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template	SOAP binding-level client assertion template for issued token SAML authentication (confirmation method bearer), with SSL message protection.
oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template	SOAP binding-level service assertion template for issued token SAML authentication (confirmation method bearer), with SSL message protection.
oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template	WS-Security 1.1 issued token SAML HOK token with certificates client assertion template. Provides authentication and message protection using Basic128.
oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template	WS-Security 1.1 issued token SAML HOK token with certificates service assertion template. Provides authentication and message protection using Basic128.
oracle/wss11_sts_issued_saml_with_message_protection_client_template	WS-Security 1.1 issued token SAML sender voucher with certificates. Provides authentication and message protection using Basic128.

C.1.4.1 oracle/sts_trust_config_client_template

The oracle/sts_trust_config_client_template invokes the STS for token exchange.

Settings

Table C–96 lists the settings for the oracle/sts_trust_config_client_template assertion template.

Table C–96 oracle/sts_trust_config_client_template Settings

Name	Description	Default Value
policy-reference-uri	The client policy URI that will be used by the client to communicate with the STS. The policy you choose depends on the authentication requirements of the STS, as identified in its WSDL.	oracle/wss10_username_token_with_message_protection_client_policy
port-endpoint	The endpoint of the STS Web service. For a WSDL 2.0 STS, the format is specified as `target-namespace#wsdl.endpoint(service-name/port-name)`. For example, `http://samples.otn.com.LoanFlow#wsdl.endpoint(LoanFlowService/LoanFlowPort` For a WSDL 1.1 STS, the format is specified as `targetnamespace#wsdl11.endpoint(servicename/portname)`. For example, `http://samples.otn.com.LoanFlow#wsdl11.endpoint(LoanFlowService/LoanFlowPort).`	None

Table C–96 (Cont.) oracle/sts_trust_config_client_template Settings

Name	Description	Default Value
port-uri	The actual endpoint URI of the STS port. For example. `http://host:port/context-root/service1`.	None
sts-keystore-recipient-alias	The alias of the STS certificate you added to the keystore. The default alias name is `sts-csf-key`.	`sts-csf-key`
wsdl-uri	The actual endpoint URI of the WSDL.	None

Configurations

Table C–97 lists the configuration properties and the default settings for the oracle/sts_trust_config_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–97 oracle/sts_trust_config_client_template Properties

Name	Description
role	SOAP role. Default settings: ■ Value—ultimateReceiver ■ Default—Not set ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.4.2 oracle/sts_trust_config_service_template

The oracle/sts_trust_config_service_template invokes the STS for token exchange.

Settings

Table C–96 lists the settings for the oracle/sts_trust_config_service_template assertion template.

Table C–98 oracle/sts_trust_config_service_template Settings

Name	Description	Default Value
port-uri	The actual endpoint URI of the STS port. For example. `http://host:port/context-root/service1`.	None
wsdl-uri	The actual endpoint URI of the WSDL.	None

Configurations

Table C–97 lists the configuration properties and the default settings for the oracle/sts_trust_config_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–99 oracle/sts_trust_config_service_template Properties

Name	Description
role	SOAP role. Default settings: ■ Value—Not set ■ Default—ultimateReceiver ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.4.3 oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template

This template inserts a SAML bearer assertion issued by a trusted STS. Messages are protected using SSL.

Settings

Table C–100 lists the settings for the oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template assertion template.

Table C–100 oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template Settings

Name	Description	Default Value
require-applies-to	Optional element in the RST. If present, Oracle WSM sends the endpoint address of the Web service for which the token is being requested. The default behavior is to always send the appliesTo element in the message from the client to the STS.	True
require-client-entropy	If a symmetric proof key is required by the Web service's security policy, the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.	Applies only to HOK.
require-server-entropy	If a symmetric proof key is required by the Web service's security policy, the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.	Applies only to HOK.
trust -version	WS-Trust version.	1.3
require-external-reference	Indicates whether external reference to the token is required.	True
require-internal-reference	Indicates whether internal reference to the token is required.	True
use-derived-keys	Indicates whether derived keys are required.	False
token-type	SAML token type. The only valid value is: 1.1.	SAML11
key-type	Key type. The only valid value is: bearer.	bearer
mutual-auth	Flag that specifies whether two-way authentication is required. Valid values include: ■ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ■ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	False
include-timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	True

Configurations

Table C–101 lists the configuration properties and the default settings for the oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–101 oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template Properties

Name	Description

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default settings:
	■ Value—Not set
	■ Default—Not set
	■ ContentType—Optional
	■ Description—Not set
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`.
	In XML format, specify an application-level map name as a `Value` for this property as follows:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

sts.auth.user.csf.key	Use to configure username/password to authenticate to the STS. If `policy-reference-uri` in the client "oracle/sts_trust_config_client_template" on page C-165 points to a username-based policy, then you configure the `sts.auth.user.csf.key` property to specify a username/password to authenticate to the STS. Default settings: ■ Value—Not set ■ ContentType—Optional
sts.auth.x509.csf.key	Use to configure X509 certificate for authenticating to the STS. If `policy-reference-uri` in the client "oracle/sts_trust_config_client_template" on page C-165 points to an x509-based policy, then you configure the `sts.auth.x509.csf.key` property to specify the X509 certificate for authenticating to the STS. Default settings: ■ Value—Not set ■ ContentType—Optional

Table C–101 (Cont.) oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template Properties

Name	Description
on.behalf.of	Optional property. Override this property to indicate whether the request is on behalf of an another entity. The default value for this flag is `false`. When set to `true` and `sts.auth.on.behalf.of.csf.key` is configured, then it will be given preference and the identity established using that CSF key will be send in the on behalf of. Otherwise, if the subject is already established, then the username from the subject will be sent as `onBehalfOf` token. If `sts.auth.on.behalf.of.csf.key` is not set and the subject does not exist, `on.behalf.of` is treated as a token exchange for the requestor and not for another entity. It is not included in an `onBehalfOf` element in the request. Default settings: ■ Value—false ■ ContentType—Optional
sts.auth.on.behalf.of.csf.key	Optional property. Use to configure on behalf of entity. If present, it will be given preference over Subject (if it exists). Default settings: ■ Value—Not set ■ ContentType—Optional
sts.auth.service.principal.name	Principal name for the Web service that needs to be protected. It is of the format `<host>/<machine name>@<REALM NAME>`. For example, `HTTP/mymachine@EXAMPLEREALM.COM`. Default settings: ■ Value—HOST/localhost@EXAMPLE.COM ■ ContentType—Optional
sts.auth.keytab.location	Location of the client's keytab file. Default settings: ■ Value—Not set ■ ContentType—Optional
sts.keystore.recipient.alias	The alias of the STS certificate you added to the keystore. The default alias name is sts-csf-key. Default settings: ■ Value—Not set ■ ContentType—Optional
sts.auth.caller.principal.name	Client's principal name as generated using the `ktpass` command and mapped to the username for which the kerberos token should be generated. It is of the format `<username>@<REALM NAME>`. Default settings: ■ Value—Not set ■ ContentType—Optional

Table C–101 (Cont.) oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template Properties

Name	Description
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

C.1.4.4 oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template

This template authenticates a SAML bearer assertion issued by a trusted STS. Messages are protected using SSL

Settings

Table C–100 lists the settings for the oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template assertion template.

Configurations

Table C–102 lists the configuration properties and the default settings for the oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–102 oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template Properties

Name	Description
role	SOAP role. Default settings: ■ Value—ultimateReceiver ■ Default—Not set ■ ContentType—Constant ■ Description—Not set
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.4.5 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template

This template inserts a SAML HOK assertion issued by a trusted STS (Security Token Service). Messages are protected using proof key material provided by the STS.

Settings

Table C–103 lists the settings for the wss11_sts_issued_saml_hok_with_message_protection_client_template assertion template.

Table C–103 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Settings

Name	Description	Default Value
require-applies-to	Optional element in the RST. If present, Oracle WSM sends the endpoint address of the Web service for which the token is being requested. The default behavior is to always send the appliesTo element in the message from the client to the STS.	True
require-client-entropy	If a symmetric proof key is required by the Web service's security policy, the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.	True
require-server-entropy	If a symmetric proof key is required by the Web service's security policy, the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.	True
trust -version	WS-Trust version.	1.3
require-external-reference	Indicates whether external reference to the token is required.	True
require-internal-reference	Indicates whether internal reference to the token is required.	True
use-derived-keys	Indicates whether derived keys are required.	False
token-type	SAML token type. The only valid values are: 1.1 and 2.0.	SAML11 and SAML20
key-type	Key type.	symmetric
is-signed	Flag that specifies whether the SAML token is signed. The only valid value for SAML policies is: `True`.	True
is-encrypted	Flag that specifies whether the SAML token is encrypted.	False
confirm-signature	Flag that specifies whether to send a signature confirmation back to the client.	True
sign-key-ref-mech	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead. This value is valid for Encryption Key Reference Mechanism only (described below.)	Thumbprint

Table C–103 (Cont.) oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template

Name	Description	Default Value
enc-key-ref-mech	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	Thumbprint
encrypt-signature	Flag that specifies whether the signature is encrypted.	False
sign-then-encrypt	Flag that specifies whether the request is signed and then encrypted.	True
algorithm-suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
include-timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	True

Configurations

Table C–104 lists the configuration properties and the default settings for the wss11_sts_issued_saml_hok_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–104 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Properties

Name	Description
sts.auth.user.csf.key	Use to configure username/password to authenticate to the STS. If `policy-reference-uri` in the client "oracle/sts_trust_config_client_template" on page C-165 points to a username-based policy, then you configure the `sts.auth.user.csf.key` property to specify a username/password to authenticate to the STS. Default settings: ▪ Value—Not set ▪ ContentType—Optional
sts.auth.x509.csf.key	Use to configure X509 certificate for authenticating to the STS. If `policy-reference-uri` in the client "oracle/sts_trust_config_client_template" on page C-165 points to an x509-based policy, then you configure the `sts.auth.x509.csf.key` property to specify the X509 certificate for authenticating to the STS. Default settings: ▪ Value—Not set ▪ ContentType—Optional

Table C–104 (Cont.) oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Properties

Name	Description
on.behalf.of	Optional property. Override this property to indicate whether the request is on behalf of an another entity. The default value for this flag is `false`. When set to `true` and `sts.auth.on.behalf.of.csf.key` is configured, then it will be given preference and the identity established using that CSF key will be send in the on behalf of. Otherwise, if the subject is already established, then the username from the subject will be sent as `onBehalfOf` token. If `sts.auth.on.behalf.of.csf.key` is not set and the subject does not exist, `on.behalf.of` is treated as a token exchange for the requestor and not for another entity. It is not included in an `onBehalfOf` element in the request.
sts.auth.on.behalf.of.csf.key	Optional property. Use to configure on behalf of entity. If present, it will be given preference over Subject (if it exists). Default settings: ■ Value—Not set ■ ContentType—Optional
sts.keystore.recipient.alias	The alias of the STS certificate you added to the keystore. The default alias name is sts-csf-key. Default settings: ■ Value—Not set ■ ContentType—Optional
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—orakey ■ ContentType—Required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–104 (Cont.) oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Properties

Name	Description

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default settings:
	■ Value—Not set
	■ Default—Not set
	■ ContentType—Optional
	■ Description—Not set
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`.
	In XML format, specify an application-level map name as a `Value` for this property as follows:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

keystore.enc.csf.key	If you set this value you then can override `keystore.enc.csf.key`, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ ContentType—Optional
sts.auth.service.principal.name	Principal name for the Web service that needs to be protected. It is of the format `<host>/<machine name>@<REALM NAME>`. For example, `HTTP/mymachine@EXAMPLEREALM.COM`. Default settings: ■ Value—Not set ■ ContentType—Optional
sts.auth.keytab.location	Location of the client's keytab file. Default settings: ■ Value—Not set ■ ContentType—Optional
sts.auth.caller.principal.name	Client's principal name as generated using the `ktpass` command and mapped to the username for which the kerberos token should be generated. It is of the format `<username>@<REALM NAME>`. Default settings: ■ Value—Not set ■ ContentType—Optional

Table C–104 (Cont.) oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Properties

Name	Description
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.4.6 oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template

This template authenticates a SAML HOK assertion issued by a trusted STS (Security Token Service). Messages are protected using WS-Security's Basic 128 suite of symmetric key technologies.

Settings

Table C–103 lists the settings for the wss11_sts_issued_saml_hok_with_message_protection_service_template assertion template.

Configurations

Table C–105 lists the configuration properties and the default settings for the wss11_sts_issued_saml_hok_with_message_protection_service_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Web Service Policies Permitting Overrides" on page 8-25.

Table C–105 oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template Properties

Name	Description
role	SOAP role. Default settings: ■ Value—ultimateReceiver ■ Default—Not set ■ ContentType—Constant ■ Description—Not set

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default settings:
	■ Value—Not set
	■ Default—Not set
	■ ContentType—Optional
	■ Description—Not set
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`.
	In XML format, specify an application-level map name as a `Value` for this property as follows:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

keystore.enc.csf.key	If you set this value you then can override keystore.enc.csf.key, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set

Table C–105 (Cont.) oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template Properties

Name	Description
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.4.7 oracle/wss11_sts_issued_saml_with_message_protection_client_template

This template inserts a SAML sender vouches assertion issued by a trusted STS (Security Token Service). Messages are protected using the client's private key.

Settings

Table C–106 lists the settings for the wss11_sts_issued_saml_with_message_protection_client_template assertion template.

Table C–106 wss11_sts_issued_saml_with_message_protection_client_template Settings

Name	Description	Default Value
require-applies-to	Optional element in the RST. If present, Oracle WSM sends the endpoint address of the Web service for which the token is being requested. The default behavior is to always send the appliesTo element in the message from the client to the STS.	True
require-client-entropy	If a symmetric proof key is required by the Web service's security policy, the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.	Applies to HOK only.
require-server-entropy	If a symmetric proof key is required by the Web service's security policy, the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.	Applies to HOK only.
trust-version	WS-Trust version.	1.3
require-external-reference	Indicates whether external reference to the token is required.	True
require-internal-reference	Indicates whether internal reference to the token is required.	True

Table C–106 (Cont.) wss11_sts_issued_saml_with_message_protection_client_template Settings

Name	Description	Default Value
use-derived-keys	Indicates whether derived keys are required.	False
token-type	SAML token type. The only valid value is: 1.1.	SAML11
is-signed	Flag that specifies whether the SAML token is signed. The only valid value for SAML policies is: `True`.	True
is-encrypted	Flag that specifies whether the SAML token is encrypted.	False
confirm-signature	Flag that specifies whether to send a signature confirmation back to the client.	True
sign-key-ref-mech	Mechanism used when signing the request. Valid values include: ▪ direct—X.509 Token is included in the request. ▪ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ▪ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ▪ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead. This value is valid for Encryption Key Reference Mechanism only (described below.)	Direct
enc-key-ref-mech	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	Thumbprint
encrypt-signature	Flag that specifies whether the signature is encrypted	False
sign-then-encrypt	Flag that specifies whether the request is signed and then encrypted.	True
algorithm-suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-191.	Basic128
include-timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	True

Configurations

Table C–107 lists the configuration properties and the default settings for the wss11_sts_issued_saml_with_message_protection_client_template assertion template. For details about the configuration property settings, see "Editing the Configuration Properties" on page 7-11.

For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

Table C–107 oracle/wss11_sts_issued_saml_with_message_protection_client_template Properties

Name	Description

csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
	Default settings:
	■ Value—Not set
	■ Default—Not set
	■ ContentType—Optional
	■ Description—Not set
	You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as the Value for this property. For example: `Value=`*`app-level-mapname`*`.map`.
	In XML format, specify an application-level map name as a `Value` for this property as follows:
	`<orawsp:Property orawsp:contentType="optional"`
	`   orawsp:name="csf.map" orawsp:type="string"/>`
	`   <orawsp:Value>app-level-mapname.map</orawsp:Value>`
	`</orawsp:Property>`
	Accessing an application-level map also requires granting credential access and identity permission to the `wsm-agent-core.jar`, as explained in "Creating an Application-level Credential Map" on page 10-24.

sts.auth.user.csf.key	Use to configure username/password to authenticate to the STS. If `policy-reference-uri` in the client "oracle/sts_trust_config_client_template" on page C-165 points to a username-based policy, then you configure the `sts.auth.user.csf.key` property to specify a username/password to authenticate to the STS. Default settings: ■ Value—Not set ■ ContentType—Optional
sts.auth.x509.csf.key	Use to configure X509 certificate for authenticating to the STS. If `policy-reference-uri` in the client "oracle/sts_trust_config_client_template" on page C-165 points to an x509-based policy, then you configure the `sts.auth.x509.csf.key` property to specify the X509 certificate for authenticating to the STS. Default settings: ■ Value—Not set ■ ContentType—Optional
on.behalf.of	Optional property. Override this property to indicate whether the request is on behalf of an another entity. The default value for this flag is `true`. When set to `true` and `sts.auth.on.behalf.of.csf.key` is configured, then it will be given preference and the identity established using that CSF key will be send in the on behalf of. Otherwise, if the subject is already established, then the username from the subject will be sent as `onBehalfOf` token. If `sts.auth.on.behalf.of.csf.key` is not set and the subject does not exist, `on.behalf.of` is treated as a token exchange for the requestor and not for another entity. It is not included in an `onBehalfOf` element in the request.

Table C–107 (Cont.) oracle/wss11_sts_issued_saml_with_message_protection_client_template Properties

Name	Description
sts.auth.on.behalf.of.csf.key	Optional property. Use to configure on behalf of entity. If present, it will be given preference over Subject (if it exists). Default settings: ■ Value—Not set ■ ContentType—Optional
sts.keystore.recipient.alias	The alias of the STS certificate you added to the keystore. The default alias name is sts-csf-key. Default settings: ■ Value—Not set ■ ContentType—Optional
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Default settings: ■ Value—orakey ■ ContentType—Optional
keystore.enc.csf.key	If you set this value you then can override keystore.enc.csf.key, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25. If you do override this value, the key for the new value must be in the keystore. That is, overriding the value does not free you from the requirement of configuring the key in the keystores. Default settings: ■ Value—Not set ■ ContentType—Optional
ignore.timestamp.in.response	Property used by the client to ignore the timestamp in the SOAP security header when it receives the response from the service. The default behavior is to *NOT* ignore the timestamp (the default value of this property is `false`). If set to `true`, then the timestamp is not required in the response message; if the timestamp is present, it is ignored. The timestamp is required to prevent replay attacks, so in general, Oracle does not recommend setting this property to `true` except to address interoperability issues. **Note:** This property is not shown in Fusion Middleware Control. Details for adding the property are described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-35.

Table C–107 (Cont.) oracle/wss11_sts_issued_saml_with_message_protection_client_template Properties

Name	Description
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.5 Authorization Assertion Templates

Table C–108 summarizes assertion templates that are used for authorization. Each authorization assertion template must follow an authentication assertion template.

Table C–108 Authorization Assertion Templates

Service Template	Description
oracle/binding_authorization_template	Provides simple role-based authorization for the request based on the authenticated subject at the SOAP binding level.
oracle/binding_permission_authorization_template	Provides simple permission-based authorization for the request based on the authenticated subject at the SOAP binding level.
oracle/component_authorization_template	Provides simple role-based authorization for the request based on the authenticated subject at the SOA component level.
oracle/component_permission_authorization_template	Provides simple permission-based authorization for the request based on the authenticated subject at the SOA component level.

C.1.5.1 oracle/binding_authorization_template

The binding_authorization_template assertion template provides simple role-based authorization for the request based on the authenticated subject at the SOAP binding level. It should follow an authentication assertion template.

Settings

Table C–109 lists the settings for the binding_authorization_template assertion template.

Table C–109 binding_authorization_template Settings

Name	Description	Default Value
Constraint Pattern	Expression that represents the constraints against which authorization checks are performed. The constraints expression is specified using the following two messageContext properties: ■ `messageContext.authenticationMethod`—Determines the authentication method used to authenticate the user. Valid value is `SAML_SV`. ■ `messageContext.requestOrigin`—Determines whether the request originated from an internal or external network. This property is valid only when using Oracle HTTP Server and the Oracle HTTP server administrator has added a custom `VIRTUAL_HOST_TYPE` header to the request. The constraint pattern properties and their values are case sensitive. The constraint expression uses the following standard supported operators: `==`, `!=`, `&&`, `||` and `!`.	
Action Pattern	Action or Web service operation for which authorization checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. For example, `validate,amountAvailable`.	actionMatchPattern
Resource Pattern	Name of the resource for which authorization checks are performed. This field accepts wildcards. For example, if the namespace of the Web service is `http://project11` and the service name is `CreditValidation`, the resource name is `http://project11/CreditValidation`.	resourceMatchPattern
Authorization Setting	Specifies the roles that are authorized. The valid values are: ■ Permit All—Permit users with any roles. ■ Deny All—Deny all users with roles. ■ Selected Roles—Permit selected roles. To add roles: 1. Click **Add**. 2. To add roles, click the checkbox next to each role you want to add in the Roles Available column and click **Move**. To add all roles, click **Move All**. To remove roles, click the checkbox next to each role you want to remove in the Roles Selected to Add column, and click **Remove**. To remove all roles, click **Remove All**. To search for roles, enter a search string in the Role Name search box and click the go arrow. The Roles Available column is updated to include only those roles that match the search string. 3. Click **OK**. To delete roles: 1. Select the role that you want to delete in the Selected Roles list. 2. Click **Delete**.	Selected Roles

Configurations

None defined.

C.1.5.2 oracle/binding_permission_authorization_template

The binding_permission_authorization_template assertion provides simple permission-based authorization for the request based on the authenticated subject at the SOAP binding level. It should follow an authentication assertion.

Settings

Table C–110 lists the settings for the binding_permission_authorization_template assertion template.

Table C–110 binding_permission_authorization_template Settings

Name	Description	Default Value
Constraint Pattern	Reserved for future use.	N/A
Action Pattern	Action or Web service operation for which permission-based checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. For example, `validate,amountAvailable`.	*
Resource Pattern	Name of the resource for which permission-based checks are performed. This field accepts wildcards. For example, if the namespace of the Web service is `http://project11` and the service name is `CreditValidation`, the resource name is `http://project11/CreditValidation`.	*
Permission Check Class	Class used for the permission-based checking. For example, `oracle.wsm.security.WSFuncPermission`.	N/A

Configurations

Table C–111 lists the configuration properties for the binding_permission_authorization_template assertion template.

Table C–111 binding_permission_authorization_template Properties

Name	Description
reference.priority	Optional property that specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. Default settings: ■ Value—Not set ■ Default—Not set ■ ContentType—Optional ■ Description—Not set The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.

C.1.5.3 oracle/component_authorization_template

The component_authorization_template assertion provides simple role-based authorization for the request based on the authenticated subject at the SOA component level. It should follow an authentication assertion.

Settings

Table C–112 lists the settings for the component_authorization_template assertion template.

Table C–112 component_authorization_template Settings

Name	Description	Default Value
Authorization Setting	Specifies the roles that are authorized. The valid values are: ■ Permit All—Permit users with any roles. ■ Deny All—Deny all users with roles. ■ Selected Roles—Permit selected roles. To add roles: 1. Click **Add**. 2. To add roles, click the checkbox next to each role you want to add in the Roles Available column and click **Move**. To add all roles, click **Move All**. To remove roles, click the checkbox next to each role you want to remove in the Roles Selected to Add column, and click **Remove**. To remove all roles, click **Remove All**. To search for roles, enter a search string in the Role Name search box and click the go arrow. The Roles Available column is updated to include only those roles that match the search string. 3. Click **OK**. To delete roles: 1. Select the role that you want to delete in the Selected Roles list. 2. Click **Delete**.	Selected Roles

Configurations

None defined.

C.1.5.4 oracle/component_permission_authorization_template

The component_permission_authorization_template assertion template provides simple permission-based authorization for the request based on the authenticated subject at the SOA component level. It should follow an authentication assertion.

> **Note:** You should be careful when using permission-based policies with EJBs as the security permissions specified in system-jazn-data.xml will be relaxed beyond a single invocation of the service operation.

Settings

Table C–113 lists the settings for the component_permission_authorization_template assertion template.

Table C–113 component_permission_authorization_template Settings

Name	Description	Default Value
Constraint Pattern	Reserved for future use.	N/A
Action Pattern	Action or Web service operation for which permission-based checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. For example, `validate,amountAvailable`.	*
Resource Pattern	Name of the resource for which permission-based checks are performed. This field accepts wildcards. For example, if the composite name of the Web service is `HelloWorld` and the service name is `Hello`, the resource name is `HelloWorld/Hello`.	*
Permission Check Class	Class used for the permission-based checking. For example, `oracle.wsm.security.WSFunctionPermission`.	N/A

Configurations

None defined.

C.1.6 Oracle Entitlements Server (OES) Integration Templates

Table C–114 summarizes the assertion templates that are used for OES integration.

Table C–114 OES Integration Templates

Service Template	Description
oracle/binding_oes_authorization_template	Sets authorization based on the policy defined in Oracle Entitlements Server (OES).
oracle/binding_oes_masking_template	Does response masking based on a policy defined in Oracle Entitlements Server (OES).
oracle/component_oes_authorization_template	Sets authorization based on the policy defined in Oracle Entitlements Server (OES). This template is used for fine-grained authorization on SCA component.

C.1.6.1 oracle/binding_oes_authorization_template

The `binding_oes_authorization_template` assertion template sets authorization based on the policy defined in Oracle Entitlements Server (OES). Authorization is based on attributes, the current authenticated subject, and the web service action invoked by the client. This template is used for fine-grained authorization on any operation on a web service. Policies based on this template should follow an authentication policy where the subject is established. Policies based on this template can be attached to any SOAP-based or REST-based endpoint.

Settings

Table C–115 lists the settings for the `binding_oes_authorization_template` assertion template.

Table C–115 binding_oes_authorization_template Settings

Name	Description	Default Value
Guard	The element defines the resource and action match values.	N/A
Action Match	Specifies the action or web service operation for which authorization checks are performed. This value can be a comma-separated list of values. This field accepts wildcards.	*
Resource Match	Specifies the name of the resource for which authorization checks are performed. This field accepts wildcards. For example, if the namespace of the web service is `http://project11` and the service name is `CreditValidation`, the resource name is `http://project11/CreditValidation`.	*

Configuration

Table C–116 lists the configuration properties and the default settings for the `binding_oes_authorization_template` assertion template.

Table C–116 binding_oes_authorization_template Configuration Properties

Name	Description	Default Value
application name	The deployed application name defined in OES. (For SOA, the composite name is used as the application name.) Value can be static or dynamic that uses ${} notation.	N/A
resource.type	Resource type defined in OES. Value can be static or dynamic that uses ${} notation. ■ For SOAP application, must be `WS_SERVICE`. ■ For SOA component, must be `COMPONENT`.	N/A
resource.name	Resource name defined in OES. Value can be static or dynamic that uses ${} notation. ■ For SOAP and SOA reference, must be of the form `web-service-name/port/web service operation`. ■ For SOA component, must be of the form `SOA component name/web service operation`.	N/A

Table C–116 (Cont.) binding_oes_authorization_template Configuration Properties

Name	Description	Default Value
lookup.action	Action that will be used during attributes lookup. Can be `request.lookup` or `response.lookup`. Value can be static or dynamic that uses ${} notation.	N/A
execute.action	Action that will be used during real authorization or masking. Default values are `authorize` for authorization and `mask` for masking use case. Value can be static or dynamic that uses ${} notation.	N/A
use.single.step	Set value to true to skip lookup phase. Does not apply to masking policy.	false
reference.priority	Optional property that specifies the priority of the policy attachment.	N/A

C.1.6.2 oracle/binding_oes_masking_template

The `oracle/binding_oes_masking_template` template does response masking based on policy defined in Oracle Entitlements Server (OES). Masking is based on attributes, current authenticated subject and web service action invoked by client. This template is used for fine-grained masking on any operation of a web service.

Settings

Table C–115 lists the settings for the `oracle/binding_oes_masking_template` assertion template.

Configuration

Table C–116 lists the configuration properties and the default settings for the `oracle/binding_oes_masking_template` assertion template.

C.1.6.3 oracle/component_oes_authorization_template

The `component_oes_authorization_template` assertion template sets user authorization based on policy defined in Oracle Entitlements Server (OES). Authorization is based on attributes, current authenticated subject and web service action invoked by client. This template is used for fine-grained authorization on SCA component.

Settings

Table C–115 lists the settings for the `component_oes_authorization_template` assertion template.

Configuration

Table C–116 lists the configuration properties and the default settings for the `component_oes_authorization_template` assertion template.

C.1.7 Supported Algorithm Suites

Table C–117 lists the algorithm suites that are supported for message protection. The algorithm suites enable you to control the cryptographic characteristics of the algorithms that are used when securing messages.

A group of standard algorithm suites are defined in WS-SecurityPolicy 1.2, which is available at the following URL:

`http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/os/ws-securitypolicy-1.3-errata01-os-complete.html#_Toc325573605`

The symmetric signature (Sym Sig) and the asymmetric signature (Asym Sig) in each suite are defaulted to `HmacSha1` and `RsaSha1` respectively as follows:

Property Algorithm	Value
[Sym Sig]	HmacSha1
[Asym Sig]	RsaSha1

OWSM also provides the extended algorithm suites as listed in the following table:

Property Algorithm	Value
[Sym Sig]	HmacSha256
[Asym Sig]	RsaSha256

The XML signatures RSA-SHA256 and HMAC-SHA256 are defined in w3c XML Security Algorithm Cross-Reference spec, which is available at the following URL:

`http://www.w3.org/TR/xmlsec-algorithms/`

Notes:

- The following algorithm suites are FIPS compliant for only wss11 policies which uses Symmetric Signature.
- For more information on FIPS, see "Enabling FIPS Mode" in *Securing Oracle WebLogic Server*.
- FIPS 140-2 support requires Oracle JDK 1.7.0_80 or higher and RSA Crypto-J V6.2.0.1 module.
- For detailed information about SSL FIPS 140-2 for OWSM, refer to support Document 2115681.1 on My Oracle Support.

Table C–117 Supported Algorithm Suites

Algorithm Suite	Digest	Encryption	Symmetric Key Wrap	Asymmetric Key Wrap	Encrypted Key Derivation	Signature Key Derivation	Minimum Signature Key Length	Symmetric Signature	Asymmetric Signature
Basic256	Sha1	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256	HmacSha1	RsaSha1
Basic192	Sha1	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1
Basic128	Sha1	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128	HmacSha1	RsaSha1
TripleDes	Sha1	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1
Basic256Rsa15	Sha1	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256	HmacSha1	RsaSha1
Basic192Rsa15	Sha1	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1
Basic128Rsa15	Sha1	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128	HmacSha1	RsaSha1
TripleDesRsa15	Sha1	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1

Note: You can follow the steps to create a new policy from a predefined policy and modify the algorithm suite using Oracle Enterprise Manager Fusion Middleware Control. For more information, see:

- "Creating Custom Policies" in *Oracle® Fusion Middleware Securing Web Services and Managing Policies with Oracle Web Services Manager*.
- "Editing a Web Service Policy" in *Oracle® Fusion Middleware Securing Web Services and Managing Policies with Oracle Web Services Manager*.

C.1.8 Message Signing and Encryption Settings for Request, Response, and Fault Messages

Table C–118 lists the settings for the Request, Response, and Fault messages. You configure these settings for message signing and encryption.

Table C–118 Request, Response, and Fault Message Signing and Encryption Settings

Name	Description	Default Value
Include Entire Body	Sign or encrypt the entire body of the SOAP message. If `false`, you can add specific body elements using the Body Elements section.	True for Request and Response messages False for Fault messages
Include SwA Attachment	Sign or encrypt SOAP messages with attachments. **Note**: This field is not applicable to MTOM attachments.	False

Table C–118 (Cont.) Request, Response, and Fault Message Signing and Encryption Settings

Name	Description	Default Value
Include MIME Headers	Sign or encrypt SOAP attachments with MIME headers. **Note**: This field is enabled and applicable if Include SwA Attachment is enabled. It is not applicable to MTOM attachments.	False
Header Elements	Sign or encrypt the specified SOAP header elements. To add a header element: **1.** Click **Add**. **2.** Enter the namespace URI. **3.** Enter the local name for the header element. **4.** Click **OK**. To edit a header element: **1.** Select the header element that you want to edit in the Header Elements list. **2.** Click **Edit**. **3.** Modify the values, as required. **4.** Click **OK**. To delete a header element: **1.** Select the header element that you want to delete in the Header Elements list. **2.** Click **Delete**. **3.** When prompted to confirm, click **OK**.	None
Body Elements	**Note**: This field is available if Include Entire Body is disabled. Sign or encrypt the specified body elements. This field is applicable if the Include Body field is disabled. To add a body element: **1.** Click **Add**. **2.** Enter the namespace URI. **3.** Enter the local name for the body element. **4.** Click **OK**. To edit a body element: **1.** Select the bpdu element that you want to edit in the Body Elements list. **2.** Click **Edit**. **3.** Modify the values, as required. **4.** Click **OK**. To delete a body element: **1.** Select the body element that you want to delete in the Body Elements list. **2.** Click **Delete**. **3.** When prompted to confirm, click **OK**.	None

C.2 Management Assertion Templates

Table C–119 summarizes the management assertion templates.

Table C–119 Management Assertion Templates

Name	Description
oracle/security_log_template	Provides a logging assertion template that can be attached to any binding or component.

C.2.1 oracle/security_log_template

The security_log_template assertion template provides a logging assertion template that can be attached to any binding or component.

> **Note:** It is recommended that the logging assertion be used for debugging and auditing purposes only.

Settings

Table C–120 lists the settings for the security_log_template assertion template.

Table C–120 security_log_template Settings

Name	Description	Default Value
Request	Requirements for logging request messages. The valid values are: ■ all—Log the entire SOAP message. ■ header—Log SOAP header information only. ■ soap_body—Log SOAP body information only. ■ soap_envelope—Log SOAP envelope information only.	all
Response	Requirements for logging response messages. The valid values are the same as for Request above.	soap_body

Configurations

None defined.

C.3 No Behavior Assertion Templates

Each of the predefined no behavior policies, described in "No Behavior Policies" on page B-41, use the same assertion that essentially does not enforce the behavior for that category.

An assertion template is not provided for this assertion. For that reason, it is important that you do not delete the no behavior policies. If you do so, you cannot recreate them and you will need to restore the repository with the original policies. For information about restoring the repository, see "Rebuilding the Oracle WSM Repository" on page 17-7.

D

Schema Reference for Predefined Assertions

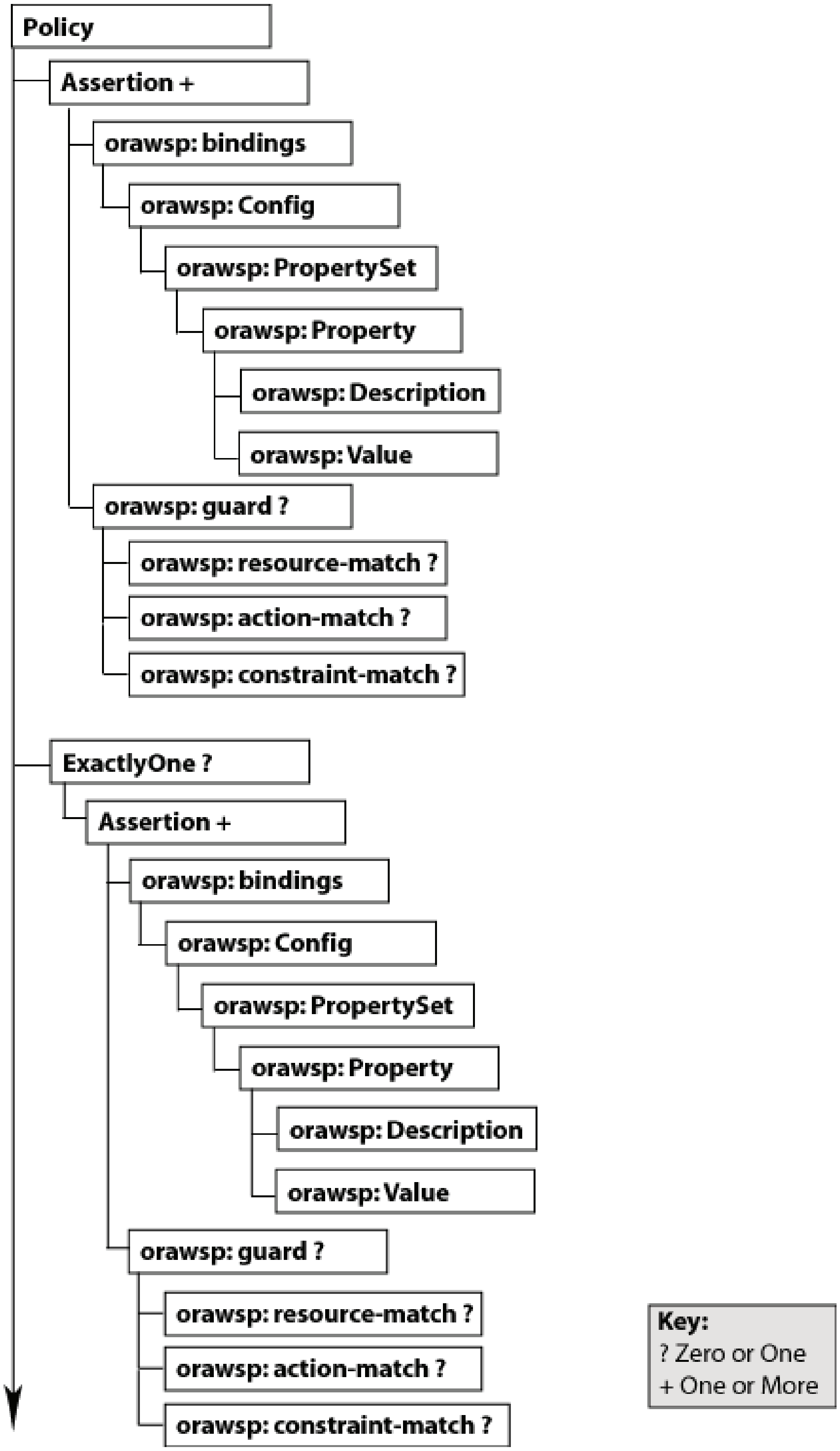
This appendix provides the XML schema for reference when creating a WS-Policy file that contains Web service assertions. Sections include:

- Graphical Representation
- Element Descriptions

D.1 Graphical Representation

The following graphic describes the element hierarchy of the assertions in the WS-Policy file.

Figure D–1 Element Hierarchy of an Assertion



The following sections describe each element and their subelements in detail:

- wsp:Policy
- wsp:ExactlyOne
- orasp:Assertion
- orawsp:bindings

- orawsp:Config
- orawsp:PropertySet
- orawsp:Property
- orawsp:Description
- orawsp:Value
- orawsp:guard
- orawsp:resource-match
- orawsp:action-match
- orawsp:constraint-match

D.2 Element Descriptions

The following sections describe the elements in the assertion in more detail. The main elements are described up front. The subelements are described following the main elements and are organized in alphabetical order.

D.2.1 wsp:Policy

Groups nested policy assertions.

D.2.1.1 Attributes

The following table summarizes the WS-Policy attributes, including the Oracle extensions.

Table D–1 Oracle Extensions to WS-Policy Attributes

Attribute	Description
Name	Name of the policy.
attachTo	Policy subjects to which the policy can be attached. Valid values include:binding.client, binding.server, binding.any.
category	Category of the policy. Valid values include: security, mtom, wsrm, addressing, and management.
description	Description of the policy.
displayName	Name displayed in the user interface.
localOptimization	Flag that specifies whether local optimization is enabled. Oracle WSM supports a SOA local optimization feature for composite-to-composite invocations in which the reference of one composite specifies a Web service binding to a second composite. Valid values include: ■ On—Local optimization is enabled ■ Off—Local optimization is turned off. The request goes through the usual WS/SOAP/HTTP process ■ Check Identity—Optimize only if a JAAS subject already exists in the current thread, indicating that authentication has already succeeded. Otherwise, go through the usual WS/SOAP/HTTP process.
status	Status of the policy reference. Valid values include: enabled and disabled.
smartDigest	Smart Digest.

Table D–1 (Cont.) Oracle Extensions to WS-Policy Attributes

Attribute	Description
oraSmartDigest	Smart Digest.
subjectCount	Number of subjects to which the policy is attached currently.
versionCreator	Author of the current version.
versionNumber	Number of the current version.
versionTime	Time the current version was creatd.
id	Policy ID.

D.2.1.2 Example

```
<wsp:Policy
 xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
 xmlns="http://schemas.xmlsoap.org/ws/2004/09/policy"
 xmlns:oralgp="http://schemas.oracle.com/ws/2006/01/loggingpolicy"
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd"
 Name="oracle/wss11_x509_token_with_message_protection_client_policy"
 orawsp:attachTo="binding.client"
 orawsp:category="security"
orawsp:description="i18n:oracle.wsm.resources.policydescription.PolicyDescription
Bundle_oracle/wss11_x509_token_with_message_protection_client_policy_PolyDescKey"
orawsp:displayName="i18n:oracle.wsm.resources.policydescription.PolicyDescription
Bundle_oracle/wss11_x509_token_with_message_protection_client_policy_
PolyDispNameKey"
 orawsp:local-optimization="check-identity"
 orawsp:oraSmartDigest="935231872"
 orawsp:smartDigest="201244603"
 orawsp:status="enabled"
 orawsp:versionCreator="mdsInternal"
 orawsp:versionNumber="1"
 orawsp:versionTime="1238006529607"
 wsu:Id="wss11_x509_token_with_message_protection_client_policy">
...
</wsp:Policy>
```

D.2.2 wsp:ExactlyOne

Optional element that defines an OR group. For more information about OR groups, see "Defining Multiple Policy Alternatives (OR Groups)" on page 3-8.

D.2.2.1 Attributes

The following table summarizes the attribute of the <wsp:ExactlyOne> element.

Table D–2 Attribute of <wsp:ExactlyOne> Element

Attribute	Description
Name	Set to OR to indicate that this is an OR group.

D.2.2.2 Example

```
<wsp:ExactlyOne orawsp:name="Or">
```

```
<orasp:wss11-saml-with-certificates orawsp:Enforced="true" orawsp:Silent="false"
   orawsp:category="security/msg-protection, security/authentication"
   orawsp:name="WS-Security 1.1 Saml  with certificates">
<orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:sign-key-ref-mech="direct"/>
<orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:confirm-signature="true" orasp:encrypt-signature="false"
   orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
   orasp:use-derived-keys="false">
...
<orasp:wss11-username-with-certificates orawsp:Enforced="true"
   orawsp:Silent="false" orawsp:category="security/authentication,
   security/msg-protection"
   orawsp:name="WS-Security 1.1 username with certificates">
<orasp:username-token orasp:add-created="false" orasp:add-nonce="false"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint"
   orasp:is-encrypted="false" orasp:is-signed="true"
   orasp:sign-key-ref-mech="thumbprint"/>
<orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:confirm-signature="true" orasp:encrypt-signature="false"
   orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
   orasp:use-derived-keys="false">
...
</wsp:ExactlyOne>
```

D.2.3 orasp:Assertion

Main element of the assertion. Valid assertion elements include:

- oralgp:Logging
- orasp:binding-authorization
- orasp:binding-permission-authorization
- orasp:coreid-security
- orasp:http-oam-security
- orasp:http-jwt-security
- orasp:http-saml20-bearer-security
- orasp:http-security
- orasp:kerberos-security
- orasp:sca-component-authorization
- orasp:sca-component-permission-authorization
- orasp:sts-trust-config
- orasp:wss10-anonymous-with-certificates
- orasp:wss10-mutual-auth-with-certificates
- orasp:wss10-saml-hok-with-certificates
- orasp:wss10-saml-token

- orasp:wss10-saml-with-certificates
- orasp:wss10-username-with-certificates
- orasp:wss11-anonymous-with-certificates
- orasp:wss11-mutual-auth-with-certificates
- orasp:wss11-saml-with-certificates
- orasp:wss11-sts-issued-token-with-certificates
- orasp:wss11-username-with-certificates
- orasp:wss-saml-token-bearer-over-ssl
- orasp:wss-saml-token-over-ssl
- orasp:wss-sts-issued-token-over-ssl
- orasp:wss-username-token
- orasp:wss-username-token-over-ssl
- rm:RMAssertion
- wsaw:UsingAddressing
- wsoma:OptimizedMimeSerialization

D.2.3.1 Attributes

The following table summarizes the attributes of the <orasp:Assertion> element.

Table D–3 Attributes of <orasp:Assertion> Element

Attribute	Description
Optional	Flag that specifies whether the assertion is optional or required.
Silent	Flag that specifies whether the assertion is advertised. If set to true, the assertion is not advertised.
Enforced	Flag that specifies whether the assertion is currently enabled. Valid values are true or false.
name	Name of the assertion.
description	Description of the assertion.
category	Category to which the assertion applies. Valid values include: security/authentication, security/msg-protection, security/authorization, security/logging, mtom, wsrm, addressing, and management.

D.2.3.2 Example

```
<orasp:wss11-mutual-auth-with-certificates orawsp:Enforced="true"
  orawsp:Silent="false" orawsp:category="security/authentication,
  security/msg-protection"
  orawsp:name="WS-Security 1.1 Mutual Auth with certificates">
...
</orasp:wss11-mutual-auth-with-certificates>
```

D.2.4 orawsp:bindings

The <oraswsp:bindings> element defines the bindings in the assertion. This element contains the following subelement:

- orawsp:Config

D.2.4.1 Example

```
<orawsp:bindings>
  <orawsp:Config orawsp:configType="declarative"
   orawsp:name="Wss11SamlWithCertsConfig">
    <orawsp:PropertySet orawsp:name="standard-security-properties">
      <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
       orawsp:type="string">
        <orawsp:Value>ultimateReceiver</orawsp:Value>
      </orawsp:Property>
    </orawsp:PropertySet>
  </orawsp:Config>
 </orawsp:bindings>
```

D.2.5 orawsp:Config

The <oraswsp:Config> element defines the configuration for the assertion. This element can contain the following subelement:

- orawsp:PropertySet

D.2.5.1 Attributes

The following table summarizes the attributes of the <orawsp:Config> element.

Table D–4 Attributes of <orawsp:Config> Element

Attribute	Description
name	Name of the configuration.
type	Category to which the configuration applies.
configType	Configuration type. Valid values include: declarative and programmatic. ■ declarative—Use deployment descriptors and configuration files to describe authentication and authorization requirements. ■ programmatic—Embed security enforcement within the application.

D.2.5.2 Example

```
<orawsp:Config orawsp:configType="declarative"
 orawsp:name="Wss11SamlWithCertsConfig">
  <orawsp:PropertySet orawsp:name="standard-security-properties">
    <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
     orawsp:type="string">
      <orawsp:Value>ultimateReceiver</orawsp:Value>
    </orawsp:Property>
  </orawsp:PropertySet>
</orawsp:Config>
```

D.2.6 orawsp:PropertySet

The <oraswsp:PropertySet> element groups nested properties. This element contains the following subelement:

- orawsp:Property

D.2.6.1 Attributes

The following table summarizes the attributes of the <orawsp:PropertySet> element.

Table D–5 Attributes of <orawsp:PropertySet> Element

Attribute	Description
name	Name of the property set.

D.2.6.2 Example

```
<orawsp:PropertySet orawsp:name="standard-security-properties">
  <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
   orawsp:type="string">
    <orawsp:Value>ultimateReceiver</orawsp:Value>
  </orawsp:Property>
</orawsp:PropertySet>
```

D.2.7 orawsp:Property

The <oraswsp:Property> element defines a single property. The following summarize valid properties used by the predefined assertions.

The <orawsp:Property> element can contain the following subelements:

- orawsp:Value

D.2.7.1 Attributes

The following table summarizes the attributes of the <orawsp:Property> element.

Table D–6 Attributes of <orawsp:Property> Element

Attribute	Description
name	Name of the property. See Table D–7 for a list of property values used by the predefined assertions.
type	Type of the property. For example, string.
contentType	Specifies whether the property is required and can be overridden. Valid values include: ■ constant—Property is a constant value and cannot be overridden. ■ required—Property is required and can be overridden. ■ optional—Property is optional and can be overridden. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-31.

The following table summarizes the properties used by the predefined assertions.

Table D–7 Properties Used by the Predefined Assertions

Property	Description
action	Action or Web service operation for which authorization checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. For example, `validate,amountAvailable`.
attesting.mapping.attribute	The mapping attribute used to represent the attesting entity. Only the DN is currently supported. This attribute is applicable only to sender vouches and then only to message protection use cases. It is not applicable to SAML over SSL policies.
audience.uri	Audience restriction.
BaseRetransmissionInterval	Interval, in milliseconds, that the source endpoint waits after transmitting a message and before it retransmits the message. If the source endpoint does not receive an acknowledgement for a given message within the interval specified by this element, the source endpoint retransmits the message. The source endpoint can modify this retransmission interval at any point during the lifetime of the sequence of messages. This assertion does not alter the formulation of messages as transmitted, only the timing of their transmission. This value defaults to 3000.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. The default value is `basic.credentials`.
csf.map	Oracle WSM map in the credential store that contains the CSF aliases.
DeliveryAssurance	Delivery assurance. Valid values include: ■ InOrder—Messages are delivered in the order they were sent. This is the default. ■ AtLeastOnce—Every message is delivered at least once. It is possible that some messages are delivered more than once. ■ AtLeastOnceInOrder—Every message is delivered at least once and in the order they were sent. It is possible that some messages are delivered more than once. ■ ExactlyOnce—Every message is delivered exactly once, without duplication. ■ ExactlyOnceInOrder—Every message is delivered exactly once, without duplication, and in the order they were sent. ■ AtMostOnce—Messages are delivered at most once, without duplication. It is possible that some messages may not be delivered at all. ■ AtMostOnceInOrder—Messages are delivered at most once, without duplication and in the order received. It is possible that some messages may not be delivered at all.
jdbc-connection-name	JNDI reference to a JDBC data store. Valid when the StoreType is set to JDBC. This value defaults to jdbc/MessagesStore.

Table D–7 (Cont.) Properties Used by the Predefined Assertions

Property	Description
InactivityTimeout	Period of inactivity (in milliseconds) for a sequence of messages. A sequence of messages is defined as a set of messages, identified by a unique sequence number, for which a particular delivery assurance applies; typically a sequence originates from a single source endpoint. If, during the duration specified by this element, a destination endpoint has received no messages from the source endpoint, the destination endpoint may consider the sequence to have been terminated due to inactivity. The same applies to the source endpoint. This value defaults to 600000.
issuer.name	Name of the JWT issuer. The default value is www.oracle.com.
keystore.enc.csf.key	The alias and password used for storing the encryption key password in the keystore. If you set this value, you then can override keystore.enc.csf.key, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25.
keystore.sig.csf.key	The alias and password used for storing the signature key password in the keystore. If you set this value, you then can override keystore.sig.csf.key, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-25
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security run time uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Can be superseded by "Using Service Identity Certification Extension".
on.behalf.of	Override this property to indicate whether the request is on behalf of an another entity. The default value for this flag is false.
permission-class	Class used for the permission-based checking. For example, `oracle.wsm.security.WSFuncPermission`.
propagate.identity.context	Propagates the identity context from the Web service client to the Web service, and then makes it available ("publishes it") to other components for authentication and authorization purposes.
realm	HTTP realm. This value defaults to owsm.
reference.priority	Specifies the priority of the policy attachment. When specified for an attached policy, the effective set of policies algorithm allows the policy with the highest integer value priority to take precedence over a conflicting policy attachment, irrespective of its scope. The value of reference.priority can be any number between(-2^{31}) and (2^{31} - 1). The higher the number, the higher the priority assigned during effective policy calculation. Any policy that does not have a value or a non-numeric value is treated as having a value of 0. If the value is set to any of the words "yes", "true", or "on", the value is set to 1. For more information, see "Specifying the Priority of a Policy Attachment" on page 9-35.
resource	Name of the resource for which authorization checks are performed. This field accepts wildcards. For example, if the namespace of the Web service is `http://project11` and the service name is `CreditValidation`, the resource name is `http://project11/CreditValidation`.
role	SOAP role. This value defaults to ultimateReceiver.
saml.assertion.filename	File containing SAML assertions. This value defaults to temp.

Table D–7 (Cont.) Properties Used by the Predefined Assertions

Property	Description
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: ■ `*` in any location. ■ `/*` at the end of the URI. ■ `.*` at the end of the URI.
saml.issuer.name	Name of the issuer of the SAML token. This value defaults to www.oracle.com.
saml.trusted.issuers	A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level.
service.principal.name	Kerberos principal name that identifies the service.
StoreName	Name of the message store. This value defaults to oracle.
StoreType	Type of message store. Valid values include: ■ InMemory—Messages are stored in memory. This is the default. ■ JDBC—Messages are stored using JDBC.
sts.auth.caller.principal.name	Client's principal name as generated using the `ktpass` command and mapped to the username for which the kerberos token should be generated. It is of the format `<username>@<REALM NAME>`.
sts.auth.keytab.location	Location of the client's keytab file.
sts.auth.on.behalf.of.csf.key	Use to configure "on behalf of" entity. If present, it will be given preference over Subject (if it exists).
sts.auth.service.principal.name	Principal name for the Web service that needs to be protected. It is of the format `<host>/<machine name>@<REALM NAME>`. For example, `HTTP/mymachine@EXAMPLEREALM.COM`.
sts.auth.user.csf.key	Use to configure username/password to authenticate to the STS. If `policy-reference-uri` in the client "oracle/sts_trust_config_client_template" on page C-165 points to a username-based policy, then you configure the `sts.auth.user.csf.key` property to specify a username/password to authenticate to the STS.
sts.auth.x509.csf.key	Use to configure X509 certificate for authenticating to the STS. If `policy-reference-uri` in the client "oracle/sts_trust_config_client_template" on page C-165 points to an x509-based policy, then you configure the `sts.auth.x509.csf.key` property to specify the X509 certificate for authenticating to the STS.
sts.keystore.recipient.alias	The alias of the STS certificate you added to the keystore. The default alias name is sts-csf-key.
subject.precedence	Set subject.precedence to false to allow for the use of a client-specified username rather than the authenticated subject. If subject.precedence is true, the user name to create the SAML assertion or JWT claim is obtained only from the Subject. Similarly, if subject.precedence is false, the user name to create the SAML assertion or JWT claim is obtained only from the csf-key username property.

Table D–7 (Cont.) Properties Used by the Predefined Assertions

Property	Description
user.attributes	Specify the attributes to be included as a comma-separated list. For example, attrib1,attrib2. The attribute names you specify must exactly match valid attributes in the configured identity store. The Oracle WSM run time reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion or JWT claim.
user.roles.include	User roles to be included in the security token. This value defaults to false.
user.tenant.name	Reserved for use with Oracle Cloud.

D.2.7.2 Example

```
<orawsp:PropertySet orawsp:name="standard-security-properties">
  <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
   orawsp:type="string">
    <orawsp:Value>ultimateReceiver</orawsp:Value>
  </orawsp:Property>
</orawsp:PropertySet>
```

D.2.8 orawsp:Description

The <oraswsp:Description> element provides a description of the property.

D.2.8.1 Example

```
<orawsp:Description>My description.</orawsp:Description>
```

D.2.9 orawsp:Value

The <oraswsp:Value> element provides a list of valid values for the property.

D.2.9.1 Example

```
<orawsp:Value>ultimateReceiver</orawsp:Value>
```

D.2.10 orawsp:guard

The <orawsp:guard> element defines the resource, action, and constraint match values.

D.2.10.1 Examples

```
<orawsp:guard>
  <orawsp:resource-match>
    http://project11/CreditValidation
  </orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>

<orawsp:guard>
  <orawsp:resource-match>*</orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
```

```
</orawsp:guard>

<orawsp:guard>
  <orawsp:constraint-match>${!(messageContext.authenticationMethod =='SAML_SV'
    || messageContext.requestOrigin == 'internal')}
  </orawsp:constraint-match>
</orawsp:guard>
```

D.2.11 orawsp:resource-match

The <orawsp:resource-match> element specifies the name of the resource for which authorization checks are performed. This field accepts wildcards.

For example, if the namespace of the Web service is `http://project11` and the service name is `CreditValidation`, the resource name is `http://project11/CreditValidation`.

D.2.11.1 Examples

```
<orawsp:guard>
  <orawsp:resource-match>
    http://project11/CreditValidation
  </orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>

<orawsp:guard>
  <orawsp:resource-match>*</orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>
```

D.2.12 orawsp:action-match

The <orawsp:resource-match> element specifies the action or Web service operation for which authorization checks are performed. This value can be a comma-separated list of values. This field accepts wildcards.

D.2.12.1 Examples

```
<orawsp:guard>
  <orawsp:resource-match>
    http://project11/CreditValidation
  </orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>

<orawsp:guard>
  <orawsp:resource-match>*</orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>
```

D.2.13 orawsp:constraint-match

The <orawsp:constraint-match> element specifies the constraints against which authorization checks are performed. The value is an expression specified using the following two messageContext properties:

- messageContext.authenticationMethod—Determines the authentication method used to authenticate the user. Valid value is `SAML_SV`.
- messageContext.requestOrigin—Determines whether the request originated from an internal or external network. This property is valid only when using Oracle HTTP Server and the Oracle HTTP server administrator has added a custom VIRTUAL_HOST_TYPE header to the request.

  The properties and their values are case sensitive. The constraint expression uses the following standard supported operators: `==`, `!=`, `&&`, `||` and `!`.

> **Note:** This element is supported with the binding-authorization element only. For other authorization assertion elements, this field is reserved for future use.

D.2.13.1 Example

```
<orawsp:guard>
<orawsp:constraint-match>${!(messageContext.authenticationMethod =='SAML_SV' ||
   messageContext.requestOrigin == 'internal')}
 </orawsp:constraint-match>
</orawsp:guard>
```

D.2.14 oralgp:Logging

The <orasp:Logging> element defines the logging policy.

The <orasp:Logging> element contains the following subelements:

- oralgp:msg-log
- orawsp:bindings

D.2.14.1 Example

```
<oralgp:Logging orawsp:Enforced="false" orawsp:Silent="true"
 orawsp:category="security/logging" orawsp:name="Log Message1">
  <oralgp:msg-log>
    <oralgp:request>all</oralgp:request>
    <oralgp:response>all</oralgp:response>
    <oralgp:fault>all</oralgp:fault>
  </oralgp:msg-log>
  <orawsp:bindings>
    <orawsp:Config orawsp:name="added-from-em"/>
  </orawsp:bindings>
</oralgp:Logging>
```

D.2.15 orasp:binding-authorization

The <orasp:binding-authorization> element defines a simple role-based authorization for the request based on the authenticated subject at the SOAP binding level.

The <orasp:binding-authorization> element contains the following subelements:

- orawsp:bindings
- orawsp:guard

It also contains **one** of the following subelements:

- orasp:denyAll

- orasp:permitAll
- orasp:role

D.2.15.1 Example

```
<orasp:binding-authorization orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authorization"
 orawsp:name="J2EE services Authorization">
  <orasp:denyAll/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="AuthzConfig"/>
  </orawsp:bindings>
  <orawsp:guard/>
</orasp:binding-authorization>
```

D.2.16 orasp:binding-permission-authorization

The <orasp:binding-permission-authorization> element defines simple permission-based authorization for the request based on the authenticated subject at the SOAP binding level.

The <orasp:binding-permission-authorization> element contains the following subelements:

- orasp:check-permission
- orawsp:bindings
- orawsp:guard

D.2.16.1 Example

```
<orasp:binding-permission-authorization orawsp:Enforced="true"
 orawsp:Silent="true" orawsp:category="security/authorization"
 orawsp:name="J2EE Permission Based Authorization">
  <orasp:check-permission/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="BindingPermissionAuthzConfig">
      <orawsp:PropertySet orawsp:name="perms-authz-properties">
        <orawsp:Property orawsp:contentType="optional" orawsp:name="resource"
         orawsp:type="string">
          <orawsp:DefaultValue>*</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional" orawsp:name="action"
         orawsp:type="string">
          <orawsp:DefaultValue>*</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="permission-class" orawsp:type="string">
          <orawsp:DefaultValue>oracle.wsm.security.WSFunctionPermission
          </orawsp:DefaultValue>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
  <orawsp:guard>
    <orawsp:resource-match>*</orawsp:resource-match>
    <orawsp:action-match>*</orawsp:action-match>
  </orawsp:guard>
```

```
</orasp:binding-permission-authorization>
```

D.2.17 orasp:coreid-security

The <orasp:coreid-security> element uses the credentials in the WS-Security header's binary security token to authenticate users against the Oracle Access Manager identity store.

It contains the following subelements:

- orasp:coreid-token
- orawsp:bindings

D.2.17.1 Example

```
<orasp:coreid-security orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authentication, security/authorization"
 orawsp:name="OAM Security">
  <orasp:coreid-token orasp:is-encrypted="false" orasp:is-signed="false"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="CoreIdConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
       </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:coreid-security>
```

D.2.18 orasp:http-jwt-security

The <orasp:http-jwt-security> element authenticates users using the username provided in the JWT token in the HTTP header.

It contains the following subelements:

- orasp:auth-header
- orasp:require-tls
- orawsp:bindings

D.2.18.1 Example

```
<orasp:http-jwt-security  orawsp:Enforced="true" orawsp:Silent="false"
orawsp:category="security/authentication" orawsp:name="Http JWT Security">
<orasp:auth-header orasp:algorithm-suite="Basic128Sha256Rsa15"
orasp:is-encrypted="false" orasp:is-signed="true" orasp:mechanism="jwt"/>
<orasp:require-tls orasp:include-timestamp="false" orasp:mutual-auth="false"/>
 <orawsp:bindings>
  <orawsp:Config orawsp:configType="declarative" orawsp:name="HttpJwtTokenConfig">
   <orawsp:PropertySet orawsp:name="standard-security-properties">
    <orawsp:Property orawsp:contentType="optional" orawsp:name="user.attributes"
orawsp:type="string"/>
    <orawsp:Property orawsp:contentType="optional" orawsp:name="issuer.name"
orawsp:type="string">
     <orawsp:Value>www.oracle.com</orawsp:Value>
    </orawsp:Property>
    <orawsp:Property orawsp:contentType="optional"
orawsp:name="user.roles.include" orawsp:type="string">
     <orawsp:Value>false</orawsp:Value>
    </orawsp:Property>
    <orawsp:Property orawsp:contentType="optional"
orawsp:name="csf.map"orawsp:type="string"/>
    <orawsp:Property orawsp:contentType="optional" orawsp:name="csf-key"
orawsp:type="string">
     <orawsp:Value>basic.credentials</orawsp:Value>
    </orawsp:Property>
    <orawsp:Property orawsp:contentType="optional"
orawsp:name="subject.precedence" orawsp:type="string">
     <orawsp:Value>true</orawsp:Value>
    </orawsp:Property>
    <orawsp:Property orawsp:contentType="optional" orawsp:name="audience.uri"
orawsp:type="string">
     <orawsp:Value/>
    </orawsp:Property>
    <orawsp:Property orawsp:contentType="optional"
orawsp:name="keystore.sig.csf.key" orawsp:type="string">
     <orawsp:Value/>
    </orawsp:Property>
    <orawsp:Property orawsp:contentType="optional"
orawsp:name="propagate.identity.context" orawsp:type="string">
     <orawsp:Value/>
    </orawsp:Property>
    <orawsp:Property orawsp:contentType="optional" orawsp:name="user.tenant.name"
orawsp:type="string">
     <orawsp:Value/>
    </orawsp:Property>
    <orawsp:Property orawsp:contentType="optional"
orawsp:name="reference.priority" orawsp:type="string"/>
   </orawsp:PropertySet>
  </orawsp:Config>
 </orawsp:bindings>
</orasp:http-jwt-security>
```

D.2.19 orasp:http-oam-security

The <orasp:http-oam-security> element uses the credentials in the HTTP header to authenticate users against the Oracle Platform Security Services identity store and propagates that information using an Oracle Access Manager (OAM) token.

It contains the following subelements:

- orasp:auth-header
- orawsp:bindings

D.2.19.1 Example

```
<orasp:http-oam-security orawsp:Silent="true" orawsp:Enforced="true"
   orawsp:name="Http Security" orawsp:category="security/authentication">
   <orasp:auth-header orasp:mechanism="oam"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:name="HttpOAMConfig"
         orawsp:configType="declarative">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:name="realm" orawsp:type="string"
               orawsp:contentType="constant">
```

```
                <orawsp:Value>owsm</orawsp:Value>
             </orawsp:Property>
             <orawsp:Property orawsp:name="role" orawsp:type="string"
                orawsp:contentType="constant">
                <orawsp:Value>ultimateReceiver</orawsp:Value>
             </orawsp:Property>
          </orawsp:PropertySet>
       </orawsp:Config>
    </orawsp:bindings>
</orasp:http-oam-security>
```

D.2.20 orasp:http-saml20-bearer-security

The <orasp:http-saml20-bearer-security> element authenticates users using credentials provided in SAML 2.0 tokens with confirmation method 'Bearer' in the HTTP header.

It contains the following subelements:

- orasp:auth-header
- orasp:require-tls
- orawsp:bindings

D.2.20.1 Example

```
<orasp:http-saml20-bearer-security
   xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
   xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
   orawsp:Enforced="true" orawsp:Silent="true"
   orawsp:category="security/authentication"
   orawsp:name="Http SAML 2.0 Bearer Security Over SSL">
   <orasp:auth-header orasp:mechanism="saml20-bearer"/>
   <orasp:require-tls orasp:include-timestamp="false"
      orasp:mutual-auth="false"/>
   <orawsp:bindings>
      <orawsp:Config orawsp:configType="declarative"
      orawsp:name="HttpSaml20BearerOverSSLConfig">
         <orawsp:PropertySet orawsp:name="standard-security-properties">
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.trusted.issuers" orawsp:type="string">
               <orawsp:Value/>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="saml.enveloped.signature.required"
               orawsp:type="boolean">
               <orawsp:Value>true</orawsp:Value>
            </orawsp:Property>
            <orawsp:Property orawsp:contentType="optional"
               orawsp:name="reference.priority" orawsp:type="string"/>
            <orawsp:Property orawsp:name="propagate.identity.context"
               orawsp:type="string" orawsp:contentType="optional">
               <orawsp:Value></orawsp:Value>
         </orawsp:PropertySet>
      </orawsp:Config>
   </orawsp:bindings>
</orasp:http-saml20-bearer-security>
```

D.2.21 orasp:http-security

The <orasp:http-security> element uses the credentials in the HTTP header to authenticate users against the Oracle Platform Security Services identity store.

It contains the following subelements:

- orasp:auth-header
- orasp:require-tls
- orawsp:bindings

D.2.21.1 Example

```
<orasp:http-security orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="Http over SSL Security">
  <orasp:auth-header orasp:mechanism="basic"/>
  <orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="HttpConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="realm"
         orawsp:type="string">
          <orawsp:Value>owsm</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:http-security>
```

D.2.22 orasp:kerberos-security

The <orasp:kerberos-security> element enforces in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

It contains the following subelements:

- orasp:kerberos-token
- orawsp:bindings
- orasp:msg-security

D.2.22.1 Example

```
<orasp:kerberos-security orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication" orawsp:name="WSS Kerberos Token">
  <orasp:kerberos-token orasp:is-encrypted="false" orasp:is-signed="false"
   orasp:type="gss-apreq-v5"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="KerberosSecurityConfig"/>
  </orawsp:bindings>
</orasp:kerberos-security>
```

D.2.23 orasp:sca-component-authorization

The <orasp:sca-component-authorization> element defines simple role-based authorization for the request based on the authenticated subject at the SOA component level.

The <orasp:sca-component-authorization> element contains the following subelement:

- orawsp:bindings

It also contains **one** of the following subelements:

- orasp:denyAll
- orasp:permitAll
- orasp:role

D.2.23.1 Example

```
<orasp:sca-component-authorization orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authorization" orawsp:name="Fabric Component
 Authorization">
  <orasp:denyAll/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="FabricAuthzConfig"/>
  </orawsp:bindings>
 </orasp:sca-component-authorization>
```

D.2.24 orasp:sca-component-permission-authorization

The <orasp:sca-component-permission-authorization> element provides simple permission-based authorization for the request based on the authenticated subject at the SOA component level.

The <orasp:binding-permission-authorization> element contains the following subelements:

- orasp:check-permission
- orawsp:bindings
- orawsp:guard

D.2.24.1 Example

```
<orasp:sca-component-permission-authorization orawsp:Enforced="true"
 orawsp:Silent="true" orawsp:category="security/authorization"
 orawsp:name="Fabric Component Authorization">
  <orasp:check-permission/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="FabricAuthzConfig">
      <orawsp:PropertySet orawsp:name="perms-authz-properties">
        <orawsp:Property orawsp:contentType="optional" orawsp:name="resource"
         orawsp:type="string">
          <orawsp:DefaultValue>*</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional" orawsp:name="action"
         orawsp:type="string">
         <orawsp:DefaultValue>*</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="permission-class" orawsp:type="string">
          <orawsp:DefaultValue>
         oracle.wsm.security.WSFunctionPermission</orawsp:DefaultValue>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
  <orawsp:guard>
    <orawsp:resource-match>*</orawsp:resource-match>
    <orawsp:action-match>*</orawsp:action-match>
  </orawsp:guard>
</orasp:sca-component-permission-authorization>
```

D.2.25 orasp:sts-trust-config

The <orasp:sts-trust-config> element provides a mechanism to invoke the STS for token exchange.

It contains the following subelements:

- orawsp:bindings

D.2.25.1 Attributes

The following table summarizes the attributes of the <orasp:sts-trust-config> element.

Table D–8 Attributes of <orasp:sts-trust-config> Element

Attribute	Description
wsdl-uri	The actual endpoint URI of the WSDL.
port-uri	The actual endpoint URI of the STS port. For example. `http://host:port/context-root/service1`.
port-endpoint	The endpoint of the STS Web service. For a WSDL 2.0 STS, the format is specified as `target-namespace#wsdl.endpoint(service-name/port-name)`. For example, `http://samples.otn.com.LoanFlow#wsdl.endpoint(LoanFlowService/LoanFlowPort)` For a WSDL 1.1 STS, the format is specified as `targetnamespace#wsdl11.endpoint(servicename/portname)`. For example, `http://samples.otn.com.LoanFlow#wsdl11.endpoint(LoanFlowService/LoanFlowPort)`.
policy-reference-uri	The client policy URI that will be used by the client to communicate with the STS. The policy you choose depends on the authentication requirements of the STS, as identified in its WSDL.
soap-version	SOAP version.
sts-keystore-recipient-alias	The alias of the STS certificate you added to the keystore. The default alias name is `sts-csf-key`.

D.2.25.2 Example

```
<orasp:sts-trust-config
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
 orasp:policy-reference-uri="oracle/wss10_username_token_with_message_protection_
client_policy"
 orasp:port-endpoint="target-namespace#wsdl.endpoint(service-name/port-name)"
 orasp:port-uri="http://host:port/sts-service" orasp:soap-version="12"
 orasp:sts-keystore-recipient-alias="sts-csf-key"
 orasp:wsdl-uri="http://host:port/sts?wsdl" orawsp:Enforced="true"
 orawsp:Silent="true" orawsp:category="security/sts-config" orawsp:name="STS
 Trust Configuration">
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative" orawsp:name="StsTrustConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
<orawsp:Property orawsp:contentType="constant" orawsp:name="role"
orawsp:type="string">
<orawsp:Value>ultimateReceiver</orawsp:Value>
</orawsp:Property>
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:sts-trust-config>
```

D.2.26 orasp:wss10-anonymous-with-certificates

The <orasp:wss10-anonymous-with-certificates> element provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

It contains the following subelements:

- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

D.2.26.1 Example

```
<orasp:wss10-anonymous-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/msg-protection"
orawsp:name="WS-Security 1.0 Anonymous with certificates">
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10AnonWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-anonymous-with-certificates>
```

D.2.27 orasp:wss10-mutual-auth-with-certificates

The <orasp:wss10-mutual-auth-with-certificates> element enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

D.2.27.1 Example

```
<orasp:wss10-mutual-auth-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/authentication,
 security/msg-protection" orawsp:name="WS-Security 1.0 Mutual Auth with
 certificates">
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10AnonWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-mutual-auth-with-certificates>
```

D.2.28 orasp:wss10-saml-hok-with-certificates

The <orasp:wss1-saml-hok-with-certificates> element provides message protection (integrity and confidentiality) and SAML holder of key based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

It contains the following subelements:

- orasp:saml-token
- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

D.2.28.1 Example

```
<orasp:wss10-saml-hok-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/authentication,
 security/msg-protection" orawsp:name="WS-Security 1.0 SAML Holder Of Key
 with certificates">
  <orasp:saml-token orasp:confirmation-type="holder-of-key"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct"
   orasp:is-encrypted="false" orasp:is-signed="true"
   orasp:rcpt-enc-key-ref-mech="direct" orasp:rcpt-sign-key-ref-mech="direct"
   orasp:sign-key-ref-mech="ski"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
        <orasp:encrypted-parts>
          <orasp:body/>
        </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10SamlHOKWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:name="keystore.recipient.alias"
         orawsp:type="string">
          <orawsp:Value>orakey</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="saml.issuer.name" orawsp:type="string">
          <orawsp:Value>www.oracle.com</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="user.roles.include" orawsp:type="string">
          <orawsp:Value>false</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
          orawsp:name="saml.assertion.filename" orawsp:type="string">
          <orawsp:Value>temp</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-saml-hok-with-certificates>
```

D.2.29 orasp:wss10-saml-token

The <orasp:wss10-saml-token> element authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.

It contains the following subelements:

- orasp:saml-token
- orawsp:bindings

D.2.29.1 Example

```
<orasp:wss10-saml-token orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication" orawsp:name="WSSecurity SAML Token">
  <orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="false" orasp:version="1.1"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssSamlTokenConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
           orawsp:type="string">
            <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-saml-token>
```

D.2.30 orasp:wss10-saml-with-certificates

The <orasp:wss10-saml-with-certificates> element enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

It contains the following subelements:

- orasp:saml-token
- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

D.2.30.1 Example

```
<orasp:wss10-saml-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/authentication,
 security/msg-protection" orawsp:name="WS-Security 1.0 SAML with certificates">
  <orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10SamlWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-saml-with-certificates>
```

D.2.31 orasp:wss10-username-with-certificates

The <orasp:wss10-username-with-certificates> element enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

It contains the following subelements:

- orasp:username-token
- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

D.2.31.1 Example

```
<orasp:wss10-username-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WS-Security 1.0 username with certificates">
  <orasp:username-token orasp:add-created="false" orasp:add-nonce="false"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
   </orasp:response>
   <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10UsernameWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-username-with-certificates>
```

D.2.32 orasp:wss11-anonymous-with-certificates

The <orasp:wss11-anonymous-with-certificates> element provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

D.2.32.1 Example

```
<orasp:wss11-anonymous-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/msg-protection"
 orawsp:name="WS-Security 1.0 Anonymous with certificates">
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss11AnonWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss11-anonymous-with-certificates>
```

D.2.33 orasp:wss11-mutual-auth-with-certificates

The <orasp:wss11-mutual-auth-with-certificates> element enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

D.2.33.1 Example

```
<orasp:wss11-mutual-auth-with-certificates orawsp:Enforced="true"
  orawsp:Silent="false" orawsp:category="security/authentication,
  security/msg-protection"
  orawsp:name="WS-Security 1.1 Mutual Auth with certificates">
  <orasp:x509-token orasp:enc-key-ref-mech="thumbprint"
   orasp:is-encrypted="false" orasp:is-signed="true"
   orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:confirm-signature="false" orasp:encrypt-signature="false"
   orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
   orasp:use-derived-keys="false">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10AnonWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:name="keystore.recipient.alias"
         orawsp:type="string">
           <orawsp:Value>orakey</orawsp:Value>
        </orawsp:Property>
     </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss11-mutual-auth-with-certificates>
```

D.2.34 orasp:wss11-saml-with-certificates

The <orasp:wss11-saml-with-certificates> element enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:saml-token

- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

D.2.34.1 Example

```
<orasp:wss11-saml-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/authentication,
 security/msg-protection" orawsp:name="WS-Security 1.1 SAML with certificates">
  <orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss11SamlWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss11-saml-with-certificates>
```

D.2.35 orasp:wss11-sts-issued-token-with-certificates

The <orasp:wss11-sts-issued-token-with-certificates> element enforces insertion of an assertion issued by a trusted STS. Messages are protected using proof key material provided by the STS, the client, or both.

It contains the following subelements:

- orasp:issued-token
- orasp:x509-token

- orasp:msg-security
- orawsp:bindings

D.2.35.1 Attributes

The following table summarizes the attributes of the <orasp:wss11-sts-issued-token-with-certificates> element.

Table D–9 Attributes of <orasp:wss11-sts-issued-token-with-certificates> Element

Attribute	Description
trust-version	WS-Trust version.
require-client-entropy	If a symmetric proof key is required by the Web service's security policy, this flag specifies whether the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.
require-server-entropy	If a symmetric proof key is required by the Web service's security policy, this flag specifies whether the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.
require-applies-to	Optional element in the RST. Flag that specifies whether Oracle WSM sends the endpoint address of the Web service for which the token is being requested. The default behavior is to always send the appliesTo element in the message from the client to the STS.

D.2.35.2 Example

```
<orasp:wss11-sts-issued-token-with-certificates
xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
orasp:require-applies-to="true" orasp:require-client-entropy="true"
orasp:require-server-entropy="true" orasp:trust-version="13"
orawsp:Enforced="true" orawsp:Silent="false"
orawsp:category="security/authentication, security/msg-protection"
orawsp:name="WS-Security 1.1, issued token">
<orasp:issued-token orasp:require-external-reference="true"
orasp:require-internal-reference="true" orasp:use-derived-keys="false">
<orasp:request-security-token-template orasp:algorithm-suite="Basic128"
orasp:key-type="Symmetric" orasp:token-type="SAML11"/>
</orasp:issued-token>
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint" orasp:is-encrypted="false"
orasp:is-signed="true" orasp:sign-key-ref-mech="thumbprint"/>
<orasp:msg-security orasp:algorithm-suite="Basic128"
orasp:confirm-signature="true" orasp:encrypt-signature="false"
orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
orasp:use-derived-keys="false">
<orasp:request>
<orasp:signed-parts>
<orasp:body/>
<orasp:header orasp:namespace="http://www.w3.org/2005/08/addressing"/>
<orasp:header orasp:namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
<orasp:header orasp:name="fmw-context"
orasp:namespace="http://xmlns.oracle.com/fmw/context/1.0"/>
</orasp:signed-parts>
<orasp:encrypted-parts>
<orasp:body/>
```

```
<orasp:header orasp:name="fmw-context"
orasp:namespace="http://xmlns.oracle.com/fmw/context/1.0"/>
</orasp:encrypted-parts>
</orasp:request>
<orasp:response>
<orasp:signed-parts>
<orasp:body/>
</orasp:signed-parts>
<orasp:encrypted-parts>
<orasp:body/>
</orasp:encrypted-parts>
</orasp:response>
<orasp:fault/>
</orasp:msg-security>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
 orawsp:name="Wss11StsIssuedTokenWithCertsConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
<orawsp:Property orawsp:contentType="optional"
 orawsp:name="sts.auth.user.csf.key" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
 orawsp:name="sts.auth.x509.csf.key" orawsp:type="string">
<orawsp:Value>enc-csf-key</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:name="on.behalf.of" orawsp:type="boolean">
<orawsp:Value>false</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
 orawsp:name="sts.auth.on.behalf.of.csf.key" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:name="keystore.recipient.alias" orawsp:type="string">
<orawsp:Value>orakey</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional" orawsp:name="keystore.enc.csf.key"
 orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
 orawsp:name="sts.auth.service.principal.name" orawsp:type="string">
<orawsp:Value>HOST/localhost@EXAMPLE.COM</orawsp:Value>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
 orawsp:name="sts.auth.keytab.location" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
<orawsp:Property orawsp:contentType="optional"
orawsp:name="sts.auth.caller.principal.name" orawsp:type="string">
<orawsp:Value/>
</orawsp:Property>
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:wss11-sts-issued-token-with-certificates>
```

D.2.36 orasp:wss11-username-with-certificates

The <orasp:wss11-username-with-certificates> element enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:username-token
- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

D.2.36.1 Example

```
<orasp:wss11-username-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WS-Security 1.1 username with certificates">
  <orasp:username-token orasp:add-created="false" orasp:add-nonce="false"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
   </orasp:response>
   <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss11UsernameWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss11-username-with-certificates>
```

D.2.37 orasp:wss-saml-token-bearer-over-ssl

The <orasp:wss-saml-token-bearer-over-ssl> element authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

It contains the following subelements:

- orasp:saml-token
- orasp:require-tls
- orawsp:bindings

D.2.37.1 Example

```
<orasp:wss-saml-token-bearer-over-ssl orawsp:Enforced="true"
 orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WSSecurity Saml Token With Confirmation method Bearer Over SSL ">
  <orasp:saml-token orasp:confirmation-type="bearer" orasp:is-encrypted="false"
   orasp:is-signed="false" orasp:version="1.1"/>
  <orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssSamlTokenBearerOverSSLConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="saml.issuer.name" orawsp:type="string">
          <orawsp:Value>www.oracle.com</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="user.roles.include" orawsp:type="string">
          <orawsp:Value>false</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss-saml-token-bearer-over-ssl>
```

D.2.38 orasp:wss-saml-token-over-ssl

The <orasp:wss-saml-token-over-ssl> element enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type.

It contains the following subelements:

- orasp:saml-token
- orasp:require-tls
- orawsp:bindings

D.2.38.1 Example

```
<orasp:wss-saml-token-over-ssl orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WSSecurity SAML Token Over SSL">
  <orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
  <orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="true"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssSamlTokenOverSSLConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="saml.issuer.name" orawsp:type="string">
          <orawsp:Value>www.oracle.com</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="user.roles.include" orawsp:type="string">
          <orawsp:Value>false</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss-saml-token-over-ssl>
```

D.2.39 orasp:wss-sts-issued-token-over-ssl

The <orasp:wss-sts-issued-token-over-ssl> element enforces authentication of a SAML assertion issued by a trusted STS. Messages are protected using SSL

It contains the following subelements:

- orasp:issued-token
- orasp:require-tls
- orawsp:bindings

D.2.39.1 Attributes

The following table summarizes the attributes of the <orasp:wss-sts-issued-token-over-ssl> element.

Table D–10 Attributes of <orasp:wss-sts-issued-token-over-ssl> Element

Attribute	Description
trust-version	WS-Trust version.
require-client-entropy	If a symmetric proof key is required by the Web service's security policy, this flag specifies whether the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.
require-server-entropy	If a symmetric proof key is required by the Web service's security policy, this flag specifies whether the requestor can pass some key material (entropy) that can be included in the calculation of the proof key. The Web service policy can indicate whether client entropy, STS entropy, or both are required.
require-applies-to	Optional element in the RST. Flag that specifies whether Oracle WSM sends the endpoint address of the Web service for which the token is being requested. The default behavior is to always send the appliesTo element in the message from the client to the STS.

D.2.39.2 Example

```
<orasp:wss-sts-issued-token-over-ssl
xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
```

```
orasp:require-applies-to="true" orasp:require-client-entropy="true"
orasp:require-server-entropy="true" orasp:trust-version="13"
orawsp:Enforced="true" orawsp:Silent="false"
orawsp:category="security/authentication, security/msg-protection"
orawsp:name="WS-Security 1.1, issued token over ssl">
<orasp:issued-token orasp:require-external-reference="true"
orasp:require-internal-reference="true" orasp:use-derived-keys="false">
<orasp:request-security-token-template orasp:key-type="Bearer"
orasp:token-type="SAML11"/>
</orasp:issued-token>
<orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
<orawsp:bindings>
<orawsp:Config orawsp:configType="declarative"
orawsp:name="WssStsIssuedTokenOverSSLConfig">
<orawsp:PropertySet orawsp:name="standard-security-properties">
<orawsp:Property orawsp:contentType="constant" orawsp:name="role"
 orawsp:type="string">
<orawsp:Value>ultimateReceiver</orawsp:Value>
</orawsp:Property>
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:wss-sts-issued-token-over-ssl>
```

D.2.40 orasp:wss-username-token

The <orasp:wss-username-token> element enforces authentication with username and password credentials in the WS-Security UsernameToken SOAP header.

It contains the following subelements:

- orasp:username-token
- orawsp:bindings

D.2.40.1 Example

```
<orasp:wss-username-token orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication"
 orawsp:name="WSSecurity UserName Token">
  <orasp:username-token orasp:add-created="false" orasp:add-nonce="false"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssUsernameTokenConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss-username-token>
```

D.2.41 orasp:wss-username-token-over-ssl

The <orasp:wss-username-token-over-ssl> element uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users against the Oracle Platform Security Services configured identity store.

It contains the following subelements:

- orasp:username-token
- orasp:require-tls
- orawsp:bindings

D.2.41.1 Example

```
<orasp:wss-username-token-over-ssl orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WSSecurity UserName Token Over SSL">
  <orasp:username-token orasp:add-created="true" orasp:add-nonce="true"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
  <orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssUsernameTokenOverSSLConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss-username-token-over-ssl>
```

D.2.42 rm:RMAssertion

The <rm:RMAssertion> element provides support for version 1.0 and version 1.1 of the Web Services Reliable Messaging protocol. The version supported depends on the XML schema namespace value used:

- WS-ReliableMessaging 1.1: http://docs.oasis-open.org/ws-rx/wsrmp/200702
- WS-ReliableMessaging 1.0: http://schemas.xmlsoap.org/ws/2005/02/rm/policy

This policy can be attached to any SOAP-based client or endpoint. Full support for this feature may require additional programming.

The <rm:RMAssertion> element contains the following subelement:

- orawsp:bindings

D.2.42.1 Example

```
<rm:RMAssertion xmlns:rm="http://schemas.xmlsoap.org/ws/2005/02/rm/policy"
  orawsp:Enforced="true" orawsp:Silent="false" orawsp:category="wsrm"
orawsp:description="i18n:oracle.wsm.resources.policydescription.PolicyDescriptionB
undle_oracle/wsrm10_policy_RMAssertion_AssertionDescKey"
 orawsp:name="RM 1.0">
  <wsp:Policy/>
  <orawsp:bindings>
    <orawsp:Config orawsp:name="RMConfig">
```

```
      <orawsp:PropertySet orawsp:name="standard-wsrm-properties">
        <orawsp:Property orawsp:name="DeliveryAssurance" orawsp:type="string">
          <orawsp:Description>Delivery Assurance. Possible values
           (case-insensitive) are InOrder,  AtLeastOnce, AtLeastOnceInOrder,
           ExactlyOnce, ExactlyOnceInOrder, AtMostOnce,
           AtMostOnceInOrder.</orawsp:Description>
          <orawsp:Value>inorder</orawsp:Value>
          <orawsp:DefaultValue>inorder</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:name="StoreType" orawsp:type="string">
          <orawsp:Description>The type of message store used. Possible values
           (case-insensitive) areInMemory, JDBC.</orawsp:Description>
          <orawsp:Value>inmemory</orawsp:Value>
          <orawsp:DefaultValue>inmemory</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:name="StoreName" orawsp:type="string">
          <orawsp:Description>The name of the message store.
          </orawsp:Description>
          <orawsp:Value>oracle</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="jdbc-connection-name" orawsp:type="string">
          <orawsp:Description>The JNDI reference to a JDBC data source, when
           the store type is JDBC.</orawsp:Description>
          <orawsp:Value>jdbc/MessagesStore</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:name="InactivityTimeout" orawsp:type="int">
          <orawsp:Description>The inactivity timeout duration, specified in
           milliseconds.</orawsp:Description>
          <orawsp:Value>600000</orawsp:Value>
         </orawsp:Property>
         <orawsp:Property orawsp:name="BaseRetransmissionInterval"
          orawsp:type="int">
           <orawsp:Description>The base retransmission interval, specified in
            milliseconds.</orawsp:Description>
           <orawsp:Value>3000</orawsp:Value>
         </orawsp:Property>
       </orawsp:PropertySet>
     </orawsp:Config>
  </orawsp:bindings>
</rm:RMAssertion>
```

D.2.43 wsaw:UsingAddressing

The <wsaw:UsingAddressing> element causes the platform to check inbound messages for the presence of WS-Addressing headers conforming to the W3C 2005 Final WS-Addressing Policy standard. In addition, it causes the platform to include a WS-Addressing header in outbound SOAP messages.

The <wsaw:UsingAddressing> element contains the following subelement:

- orawsp:bindings

D.2.43.1 Example

```
<wsaw:UsingAddressing xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
 orawsp:Enforced="true" orawsp:Silent="false" orawsp:category="addressing"
 orawsp:name="WS-Addressing 2005">
  <orawsp:bindings>
    <orawsp:Config orawsp:name="added-from-em"/>
  </orawsp:bindings>
</wsaw:UsingAddressing>
```

D.2.44 wsoma:OptimizedMimeSerialization

The <wsoma:OptimizedMimeSerialization> element rejects inbound messages that are not in MTOM format and verifies that outbound messages are in MTOM format. MTOM refers to specifications `http://www.w3.org/TR/2005/REC-soap12-mtom-20050125` and `http://www.w3.org/Submission/2006/SUBM-soap11mtom10-20060405` for SOAP 1.2 and SOAP 1.1 bindings, respectively.

The <wsoma:OptimizedMimeSerialization> element contains the following subelement:

- orawsp:bindings

D.2.44.1 Example

```
<wsoma:OptimizedMimeSerialization
 xmlns:wsoma=
 "http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization"
 orawsp:Enforced="true" orawsp:Silent="false" orawsp:category="mtom"
 orawsp:name="MTOM">
  <orawsp:bindings>
    <orawsp:Config orawsp:name="added-from-em"/>
  </orawsp:bindings>
</wsoma:OptimizedMimeSerialization>
```

D.2.45 oralgp:fault

The <oralgp:fault> element configures logging for the fault message. Valid values include:

- all—Log the entire SOAP message.
- header—Log SOAP header information only.
- soap_body—Log SOAP body information only.
- soap_envelope—Log SOAP envelope information only.

D.2.45.1 Example

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

D.2.46 oralgp:request

The <oralgp:request> element configures logging for the request message. Valid values include:

- all—Log the entire SOAP message.
- header—Log SOAP header information only.

- soap_body—Log SOAP body information only.
- soap_envelope—Log SOAP envelope information only.

D.2.46.1 Example

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

D.2.47 oralgp:response

The <oralgp:response> element configures logging for the response message. Valid values include:

- all—Log the entire SOAP message.
- header—Log SOAP header information only.
- soap_body—Log SOAP body information only.
- soap_envelope—Log SOAP envelope information only.

D.2.47.1 Example

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

D.2.48 oralgp:msg-log

The <oralgp:msg-log> element configures logging for the request, response, and fault messages. The <oralgp:msg-log> element contains the following subelements:

- oralgp:request
- oralgp:response
- oralgp:fault

D.2.48.1 Example

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

D.2.49 orasp:attachment

The <orasp:attachment> element defines the attachment information.

D.2.49.1 Attributes

The following table summarizes the attributes of the <orasp:attachment> element.

Table D–11 Attributes of <orasp:attachment> Element

Attribute	Description
include-mime-headers	Flag that specifies whether or include MIME headers. Valid values include true or false.

D.2.49.2 Example

```
<orasp:signed-parts>
  <orasp:header orasp:name="From"
    orasp:namespace="http://www.w3.org/2005/08/addressing"/>
   <orasp:attachment orasp:include-mime-headers="false"/>
</orasp:signed-parts>
```

D.2.50 orasp:auth-header

The <orasp:auth-header> element specifies the name of the authentication header.

D.2.50.1 Attributes

The following table summarizes the attribute of the <orasp:auth-header> element.

Table D–12 Attributes of <orasp:auth-header> Element

Attribute	Description
algorithm-suite	Algorithm suite used to sign security tokens. For example, Basic128Sha256Rsa15. For more information, see "Supported Algorithm Suites" on page C-191.
is-encrypted	Flag that specifies whether the security token is encrypted. Valid values include true or false.
is-signed	Flag that specified whether the security token is signed. Valid values include true or false.
mechanism	Authentication mechanism. Valid values include: ▪ basic—Client authenticates itself by transmitting the username and password. ▪ cert—**Not supported in this release**Client authenticates itself by transmitting a certificate. ▪ custom—**Not supported in this release**Custom authentication mechanism. ▪ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ▪ jwt—Client authenticates itself using JWT token. ▪ oam—Client authenticates itself using OAM agent. ▪ saml20-bearer—Client authenticates itself using SAML 2.0 Bearer token. ▪ spnego—Client authenticates itself using Kerberos SPNEGO.

D.2.50.2 Examples

```
<orasp:auth-header orasp:mechanism="basic"/>
```

```
<orasp:auth-header orasp:algorithm-suite="Basic128Sha256Rsa15"
orasp:is-encrypted="false" orasp:is-signed="true" orasp:mechanism="jwt"/>
```

D.2.51 orasp:body

The <orasp:body> element defines the message body elements that are signed and encrypted. To include the entire body, specify the body element as follows: <orasp:body/>.

D.2.51.1 Example

```
  <orasp:request>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:request>
```

D.2.52 orasp:check-permission

The <orasp:check-permission> element specifies that permissions are to be checked.

D.2.52.1 Example

```
<orasp:binding-permission-authorization orawsp:Enforced="true"
 orawsp:Silent="true" orawsp:category="security/authorization"
 orawsp:name="J2EE Permission Based Authorization">
  <orasp:check-permission/>
  ...
</orasp:binding-permission-authorization>
```

D.2.53 orasp:coreid-token

The <orasp:coreid-token> element defines the OAM token.

D.2.53.1 Attributes

The following table summarizes the attributes of the <orasp:coreid-token> element.

Table D–13 Attributes of <orasp:coreid-token> Element

Attribute	Description
is-encrypted	Flag that specifies whether the assertion is encrypted. Valid values include true or false.
is-signed	Flag that specifies whether the assertion is signed. Valid values include true or false.

D.2.53.2 Example

```
<orasp:coreid-token orasp:is-encrypted="false" orasp:is-signed="false"/>
```

D.2.54 orasp:denyAll

The <orasp:denyAll> element denies all users with any roles.

D.2.54.1 Example

```
<orasp:binding-authorization orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authorization"
 orawsp:name="J2EE services Authorization">
  <orasp:denyAll/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="AuthzConfig"/>
  </orawsp:bindings>
  <orawsp:guard/>
</orasp:binding-authorization>
```

D.2.55 orasp:element

The <orasp:element> element defines a header or body element that is signed or encrypted.

D.2.55.1 Attributes

The following table summarizes the attributes of the <orasp:element> element.

Table D–14 Attributes of <orasp:element> Element

Attribute	Description
name	Name of the header or body element.
namespace	Namespace.

D.2.55.2 Example

```
<orasp:signed-elements>
  <orasp:element orasp:name="BodyElement"
   orasp:namespace="http://www.w3.org/2005/08/addressing">n/a</orasp:element>
</orasp:signed-elements>
```

D.2.56 orasp:encrypted-elements

The <orassp:encrypted-elements> element defines the message body elements that are signed. This element is valid if <orasp:encrypted-parts> is not set to <orasp:body/>

The <orassp:encrypted-parts> element contains the following subelement:

- orasp:element

D.2.56.1 Example

```
<orasp:encrypted-elements>
  <orasp:element orasp:name="Myhead"
   orasp:namespace="http://www.w3.org/2005/08/addressing">n/a</orasp:element>
</orasp:encrypted-elements>
```

D.2.57 orasp:encrypted-parts

The <orasp:encrypted-parts> element defines the message parts that are encrypted.

The <orasp:encrypted-parts> element contains one or more of the following subelements:

- orasp:body

- orasp:header
- orasp:attachment

D.2.57.1 Example

```
<orasp:request>
  <orasp:signed-parts>
    <orasp:body/>
  </orasp:signed-parts>
  <orasp:encrypted-parts>
    <orasp:body/>
  </orasp:encrypted-parts>
</orasp:request>
```

D.2.58 orasp:fault

The <orasp:fault> element defines the message body elements that are signed and encrypted in the fault message. The <orasp:fault> element contains the following subelements:

- orasp:signed-parts
- orasp:encrypted-parts

D.2.58.1 Example

```
<orasp:response>
  <orasp:signed-parts>
    <orasp:body/>
  </orasp:signed-parts>
  <orasp:encrypted-parts>
    <orasp:body/>
  </orasp:encrypted-parts>
</orasp:response>
```

D.2.59 orasp:header

The <orasp:header> element defines a header element.

D.2.59.1 Attributes

The following table summarizes the attributes of the <orasp:header> element.

Table D–15 Attributes of <orasp:header> Element

Attribute	Description
name	Name of the header element. The default header elements in the predefined namespace include: To, From, FaultTo, ReplyTo, MessageID, RelatesTo, and Action.
namespace	Namespace. The predefined namespace is as follows: http://www.w3.org/2005/08/addressing.

D.2.59.2 Example

```
<orasp:signed-parts>
  <orasp:header orasp:name="From"
    orasp:namespace="http://www.w3.org/2005/08/addressing"/>
   <orasp:attachment orasp:include-mime-headers="false"/>
```

```
</orasp:signed-parts>
```

D.2.60 orasp:issued-token

The <orasp:issued-token> element enforces token characteristics.

D.2.60.1 Attributes

The following table summarizes the attributes of the <orasp:issued-token> element.

Table D–16 Attributes of <orasp:issued-token> Element

Attribute	Description
use-derived-keys	Flag that specifies whether derived keys are required. Possible values are True and False.
require-internal-reference	Flag that specifies whether internal reference to the token is required. Possible values are True and False.
require-external-reference	Flag that specifies whether external reference to the token is required. Possible values are True and False.

D.2.60.2 Example

```
<orasp:issued-token orasp:require-external-reference="true"
 orasp:require-internal-reference="true" orasp:use-derived-keys="false">
```

D.2.61 orasp:kerberos-token

The <orasp:kerberos-token> element defines the kerberos token.

D.2.61.1 Attributes

The following table summarizes the attributes of the <orasp:kerberos-token> element.

Table D–17 Attributes of <orasp:kerberos-token> Element

Attribute	Description
is-encrypted	Flag that specifies whether the assertion is encrypted. Valid values include true or false.
is-signed	Flag that specifies whether the assertion is signed. Valid values include true or false.
type	Type of Kerberos token. The only valid value is gss-apreq-v5 (Kerberos Version 5 GSS-API).

D.2.61.2 Example

```
<orasp:kerberos-token orasp:is-encrypted="false" orasp:is-signed="false"
 orasp:type="gss-apreq-v5"/>
```

D.2.62 orasp:msg-security

The <orassp:msg-security> element defines message security for the policy. You define the body elements that are signed and encrypted for the request, response, and fault.

The <orasp:msg-security> element contains the following subelements:

- orasp:request
- orasp:response
- orasp:fault

D.2.62.1 Attributes

The following table summarizes the attributes of the <orasp:msg-security> element.

Table D–18 Attributes of <orasp:msg-security> Element

Attribute	Description
algorithm-suite	Defines the algorithm suite that is used for message protection. For example, Basic128. For more information, see "Supported Algorithm Suites" on page C-191.
confirm-signature	Flag that specifies whether to send a signature confirmation back to the client. Valid values inlcude true or false.
encrypt-signature	Flag that specifies whether to send a encryption confirmation back to the client. Valid values inlcude true or false.
include-timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.
sign-then-encyrpt	Flag that specifies whether to sign the message before encrypting the message.
use-derived-keys	Flag that specifies whether to use derived keys.

D.2.62.2 Example

```
<orasp:msg-security orasp:algorithm-suite="Basic128"
orasp:confirm-signature="false" orasp:encrypt-signature="false"
orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
orasp:use-derived-keys="false">
  <orasp:request>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:request>
  <orasp:response>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:response>
  <orasp:fault/>
</orasp:msg-security>
```

D.2.63 orasp:permitAll

The <orasp:permitAll> element permits all users with any roles.

D.2.63.1 Example

```
<orasp:binding-authorization orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authorization"
 orawsp:name="J2EE services Authorization">
  <orasp:permitAll/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="AuthzConfig"/>
  </orawsp:bindings>
</orasp:binding-authorization>
```

D.2.64 orasp:request

The <orasp:request> element defines the message body elements that are signed and encrypted in the request message. The <orasp:request> element contains the following subelements:

- orasp:signed-parts
- orasp:encrypted-parts

D.2.64.1 Example

```
  <orasp:request>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:request>
```

D.2.65 orasp:require-tls

The <orasp:require-tls> element specifies whether two-way authentication is required.

D.2.65.1 Attributes

The following table summarizes the attributes of the <orasp:require-tls> element.

Table D–19 Attributes of <orawsp:require-tls> Element

Attribute	Description
include-timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid. **Note**: This attribute is not valid for RESTful servlet applications.
mutual-auth	Flag that specifies whether two-way authentication is required. Valid values include true or false.

D.2.65.2 Examples

```
<orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
```

D.2.66 orasp:response

The <orasp:response> element defines the message body elements that are signed and encrypted in the response message. The <oraswsp:response> element contains the following subelements:

- orasp:signed-parts
- orasp:encrypted-parts

D.2.66.1 Example

```
<orasp:response>
  <orasp:signed-parts>
    <orasp:body/>
  </orasp:signed-parts>
  <orasp:encrypted-parts>
    <orasp:body/>
  </orasp:encrypted-parts>
</orasp:response>
```

D.2.67 orasp:role

The <orasp:role> element defines the roles that are permitted access.

D.2.67.1 Attribute

The following table summarizes the attribute of the <orasp:role> element.

Table D–20 Attributes of <orasp:role> Element

Attribute	Description
name	Name of the role. Valid roles include: ■ Monitor ■ AdminChannelUsers ■ Administrators ■ OracleSystemGroup ■ Operators ■ CrossDomainConnectors ■ Deployers ■ AppTesters

D.2.67.2 Example

```
<orasp:binding-authorization orawsp:Enforced="true" orawsp:Silent="true"
  orawsp:category="security/authorization" orawsp:description=""
  orawsp:name="J2EE services Authorization">
  <orasp:role orasp:name="Monitors"/>
  <orasp:role orasp:name="AdminChannelUsers"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="AuthzConfig"/>
  </orawsp:bindings>
</orasp:binding-authorization>
```

D.2.68 orasp:saml-token

The <orasp:saml-token> element configures the SAML token.

D.2.68.1 Attributes

The following table summarizes the attributes of the <orasp:saml-token> element.

Table D–21 Attributes of <orasp:saml-token> Element

Attribute	Description
confirmation-type	Confirmation type. Valid values include: sender-vouches and holder-of-key. ■ sender-vouches ■ holder-of-key ■ bearer
is-encrypted	Flag that specifies whether the assertion is encrypted. Valid values include true or false.
is-signed	Flag that specifies whether the assertion is signed. Valid values include true or false.
version	SAML version. Valid values include: 1.1 and 2.0.

D.2.68.2 Example

```
<orasp:saml-token orasp:confirmation-type="holder-of-key"
 orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
```

D.2.69 orasp:signed-elements

The <orassp:signed-elements> element defines the message body elements that are signed. This element is valid if <orasp:signed-parts> is not set to <orasp:body/>

The <orassp:signed-elements> element contains the following subelement:

- orasp:element

D.2.69.1 Example

```
<orasp:signed-elements>
  <orasp:element orasp:name="Myhead"
   orasp:namespace="http://www.w3.org/2005/08/addressing">n/a</orasp:element>
</orasp:signed-elements>
```

D.2.70 orasp:signed-parts

The <orasp:signed-parts> element defines the message parts that are signed.

The <orasp:signed-parts> element contains one or more of the following subelements:

- orasp:body
- orasp:header
- orasp:attachment

D.2.70.1 Example

```
  <orasp:request>
```

```
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:request>
```

D.2.71 orasp:username-token

The <orasp:username-token> element configures the SAML token.

D.2.71.1 Attributes

The following table summarizes the attributes of the <orasp:username-token> element.

Table D–22 Attributes of <orasp:username-token> Element

Attribute	Description
add-created	Flag that specifies whether a time stamp for the creation of the username token is required. **Note**: If Password Type is set to digest, then this attribute must be set to true. Otherwise, the policy to which it is attached will not validate.
add-nonce	Flag that specifies whether a nonce must be included with the username to prevent replay attacks. **Note**: If Password Type is set to digest, then this attribute must be set to true. Otherwise, the policy to which it is attached will not validate.
is-encrypted	Flag that specifies whether the username is encrypted. Valid values include true or false.
is-signed	Flag that specifies whether the username is signed. Valid values include true or false.
password-type	Type of password required. Valid values are: ■ none—No password. ■ plaintext—Unencrypted password in clear text. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest.

D.2.71.2 Example

```
<orasp:username-token
  orasp:add-created="false"
  orasp:add-nonce="false"
  orasp:is-encrypted="true"
  orasp:is-signed="true"
  orasp:password-type="plaintext"/>
```

D.2.72 orasp:x509-token

The <orasp:x509-token> element defines the x.509 digital certificate.

D.2.72.1 Attributes

The following table summarizes the attributes of the <orasp:x509-token> element.

Table D–23 Attributes of <orasp:x509-token> Element

Attribute	Description
sign-key-ref-mech	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead. This value is valid for Encryption Key Reference Mechanism only (described below.)
enc-key-ref-mech	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.
rcpt-sign-key-ref-mech	Mechanism used when signing the receipt. Valid values are the same as for Sign Key Reference Mechanism above.
rcpt-enc-key-ref-mech	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.
is-encrypted	Flag that specifies whether the assertion is encrypted. Valid values include true or false.
is-signed	Flag that specifies whether the assertion is signed. Valid values include true or false.

D.2.72.2 Example

```
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint"
 orasp:is-encrypted="false" orasp:is-signed="true"
 orasp:sign-key-ref-mech="direct"/>
```

D.2.73 orawsp:Description

The <oraswsp:Description> element provides a description of the property.

D.2.73.1 Example

```
<orawsp:Description>Valid IP Values</orawsp:Description>
```

E

Schema Reference for Policy Sets

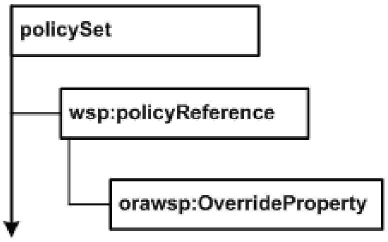
This appendix provides the XML schema for reference when creating a policy set file. Sections include:

- Graphical Representation
- Element Descriptions

E.1 Graphical Representation

The following graphic describes the element hierarchy of the policy set document.

Figure E–1 Element Hierarchy of the Policy Set



The following sections describe each element and their attributes in more detail.

E.2 Element Descriptions

This section describes the policy set elements.

E.2.1 policySet

A policy set is used to define a set of concrete policies that apply to some binding type or implementation type. Physically, a policy set is expressed as an XML element using the pseudo-schema shown in Example E–1.

E.2.1.1 Attributes

The following section summarizes the policy set attributes, including the Oracle extensions.

Table E–1 Attributes of Policy Set Element

Attribute	Description
name	Name of the policy set.

Table E–1 (Cont.) Attributes of Policy Set Element

Attribute	Description
appliesTo	Supported expression identifying an element to which the policy set applies. This attribute must contain a value to be considered valid.
attachTo	Supported expression identifying an element to which the policy set is attached. This attribute must contain a value to be considered valid.
description	Description for the policy set. This name is used when the policy set is displayed in a user interface.
status	Indicates if a policy set is available for use. When set to enabled (the default), the policy set is processed normally. When set to disabled, the policy set is ignored during processing. This attribute is automatically set to disabled if the policy set fails validation when written to the repository.
constraint	Supported expression identifying the run-time context to which the policy set applies. If this attribute is not specified, the policy set applies to all run-time contexts.

E.2.2 wsp:policyReference

Element used to associate a policy set with one or more policies. This element contains the following subelement:

- orawsp:OverrideProperty

E.2.2.1 Attributes

The following table summarizes the attributes of the <wsp:policyReference> element.

Table E–2 Attributes of <wsp:policyReference> Element

Attribute	Description
URI	Oracle WSM policy URI to be associated with the policy set.
category	Category of the policy. Valid values include: security, mtom, wsrm, addressing, and management.
status	Status of the policy reference. Valid values include: enabled and disabled.

E.2.2.2 Example

The following example illustrates a sample policy set that attaches a username token policy to all non-SCA web services in an application whose name begins with the text "CRM" in a domain named "base_domain".

Example E–1 Sample policySet Element

```
<policySet name="non_sca_web_service_policyset"
           appliesTo="WS_Service()"
           attachTo="Domain('base_domain') and Application('CRM*')"
           orawsp:description="Default policy for a non-SCA web service"
           orawsp:status="enabled"
           xmlns="http://docs.oasis-open.org/ns/opensca/sca/200903"
           xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
           xmlns:wsp="http://www.w3.org/ns/ws-policy">
```

```
    <wsp:PolicyReference
        wsp:URI="oracle/wss_username_token_service_policy"
        orawsp:category="security"
        orawsp:status="enabled" />
</policySet>
```

E.2.3 orawsp:OverrideProperty

The <orawsp:OverrideProperty> element is used to specify a configuration override associated with a policy attachment in a policy set.

E.2.3.1 Example

```
<orawsp:OverrideProperty name="csf-key" value="orakey" />
```